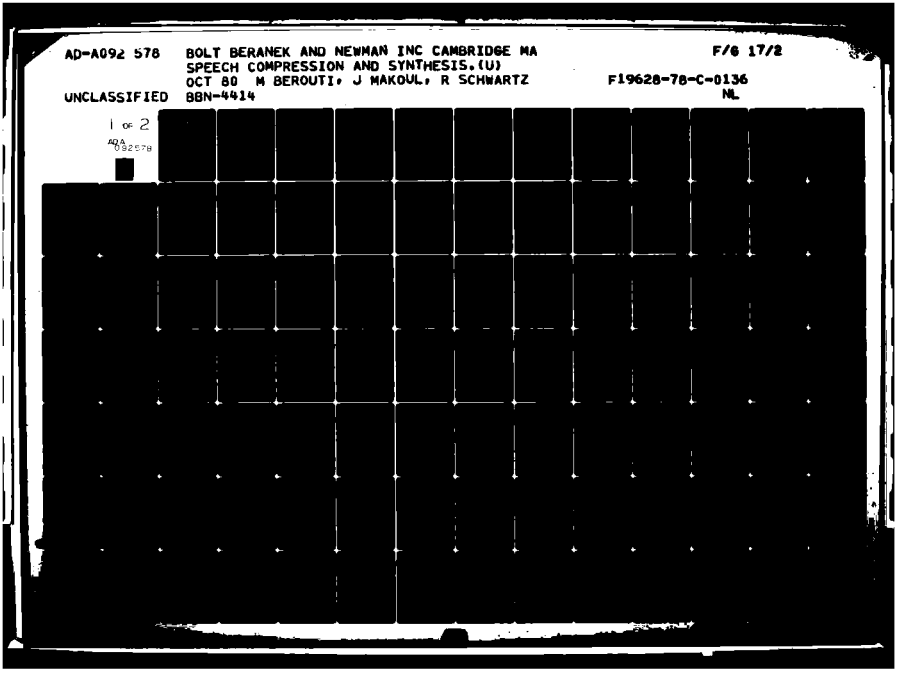
AD-A092 578 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
SPEECH COMPRESSION AND SYNTHESIS.(U)
OCT 80 M BEROUTI, J MAKOUL, R SCHWARTZ F19628-78-C-0136
UNCLASSIFIED BBN-4414 NL

1 OF 2
AD A092578



Bolt Beranek and Newman Inc.

Report No. 4414

Speech Compression and Synthesis

Final Report
Contract No. F19628-78-C-0136

October 1980

Submitted to:

Advanced Research Projects Agency
and RADC/EEV

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 4414	2. GOVT ACCESSION NO.: AD-A092578
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): SPEECH COMPRESSION AND SYNTHESIS.	5. TYPE OF REPORT & PERIOD COVERED: Final Report
	6. PERFORMING ORG. REPORT NUMBER: Report No. 4414
7. AUTHOR(s): Michael Berouti, Richard Schwartz, John Makhoul, John Sorensen	8. CONTRACT OR GRANT NUMBER(s): F19628-78-C-0136
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc., 50 Moulton Street, Cambridge, MA	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS:	12. REPORT DATE: October 1980
	13. NUMBER OF PAGES: 109
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Deputy for Electronic Technology (RADC/ETC), Hanscom Air Force Base, MA 01731, Contract Monitor: Mr. Caldwell P. Smith	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3515.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech synthesis, phonetic synthesis, diphone, LCP synthesis, vocoder, speech compression, linear prediction, voice-excited coder, high-frequency regeneration, spectral duplication, real-time vocoder, mixed-source excitation model, adaptive lattice, phonetic vocoder, spectral template,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report concludes our work for the past two years on speech compression and synthesis. Since there was an annual report for the first year, this report concentrates on reporting the work for the second year. In the first year we implemented a real-time variable-frame-rate LPC vocoder operating at an average rate of 2000 bits/s. We also tested our mixed-source model as part of the vocoder. To improve the reliability of the extraction of LPC parameters, we implemented and tested a range of adaptive lattice and autocorrelation algorithms. For data rates above 5000 bits/s,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

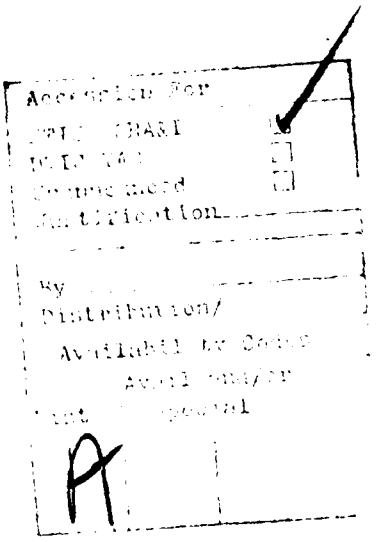
Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

continued from Block 19:
speech recognition, phoneme recognition.

continued from Block 20:
we developed and tested a new high-frequency regeneration technique, spectral duplication, which reduces the roughness in the synthesized speech. As the first part of our effort towards a very-low-rate (VLR) vocoder, we implemented a phonetic synthesis program that would be compatible with our initial design for a phonetic recognition program. We also recorded and partially labeled a large data base of diphone templates.

During the second year we continued our work toward a VLR vocoder, and also developed a multirate embedded-coding speech compression program that could transmit speech at rates varying from 9600 to 2400 b/s. The phonetic synthesis program and the labeling of the diphone template network were completed. There are currently 2845 diphone templates. We also implemented an initial version of a phonetic recognizer based on a network representation of diphone templates. The recognizer allows for incremental training of the network by modification of existing templates or addition of new templates.



Report No. 4414

SPEECH COMPRESSION AND SYNTHESIS

October 1980

Prepared by:

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02238

Submitted to:

Advanced Research Projects Agency
and RADC/EEV

Report No. 4414 Bolt Beranek and Newman Inc.

SPEECH COMPRESSION AND SYNTHESIS

Final Report

ARPA Order No. 3515

Contract No. F19628-78-C-0136

Name of Contractor:
Bolt Beranek and Newman Inc.

Principal Investigators:
Dr. John Makhoul
(617) 491-1850, x4332

Dr. R. Viswanathan
(617) 491-1850 x4336

Effective Date of Contract:
6 April 1978

Contract Expiration Date:
31 July 1980

Sponsored by
Defense Advanced Research Projects Agency (DoD)
Monitored by RADC/EEV

PROJECT PERSONNEL

John Makhoul	Principal Scientist
A.W.F. (Bill) Huggins	Senior Scientist
R. (Vishu) Viswanathan	Senior Scientist
Jared Wolf	Senior Scientist
Michael Berouti	Scientist
John Klovstad	Scientist
Richard Schwartz	Scientist
John Sorensen	Staff Scientist
Victor Zue	Senior Consultant
Nancy Chapman	Project Secretary

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In this final report we present our work performed during the period June 8, 1979 to July 31, 1980 in the area of speech compression and synthesis. The reader is referred to the previous annual report [1] for work performed between April 6, 1978 and June 7, 1979 under this contract.

In Section 1.1 we give a very brief list of the major accomplishments in the past year. The reader is referred to the body of the report for details on these as well as other accomplishments. An outline of this report is given in Section 1.2. In Section 1.3, we give a list of the presentations and publications for past year. The publications are included in the Appendix.

1.1 Major Accomplishments

a) Completed the labelling of the diphone data base for phonetic synthesis. The total number of diphones is now 2733.

   Generalized the phonetic synthesis program to allow the use of phonological rules combined with diphone templates.

   Improved the algorithms used by the phonetic synthesis program for gain normalization and time warping.

b) Wrote program to interface the MITALK text-to-speech program to our diphone synthesis program.

c) Developed the Harmonic Deviations Vocoder for LPC analysis/synthesis. This new method models the speech spectrum more accurately by compensating for the spectral errors in the LPC spectrum at the pitch harmonics.

d) Designed and implemented a phonetic recognition program that compares input speech to a network of diphone templates. This program produces a sequence of phonemes, durations and pitch values suitable for input to the phonetic synthesis program. An initial network has been constructed from the diphone template data base. The program also allows the user to augment the diphone network interactively.

e) Implemented extended addressing in the BCPL compiler and our user programs under the KL-10 TOPS20-Release 4 monitor. This allows data structures of up to 8 million words to be addressable by a single user process. This improvement enables the entire diphone network to be "in core" at all times.

f) Implemented and tested an embedded-code multirate adaptive transform coding system capable of operating at an arbitrary data rate in the range 2.5 to 9.6 kb/s.

1.2 Outline

In Section 2 we describe this year's work on the phonetic synthesis part of the very-low-rate (VLR) phonetic vocoder. Our initial design and implementation of the phonetic recognition part of the phonetic vocoder is discussed in Section 3. Section 4 contains a description of our embedded-code multirate adaptive transform coding system.

1.3 Presentations and Publications

During the past year, we gave a number of oral presentations at the regular ARPA Network Speech Compression (NSC) Meetings. In addition, we made five presentations at conferences and had one paper published. These were:

a) M. Berouti and J. Makhoul, "An Adaptive-Transform Baseband Coder," Proceedings of the 97th Meeting of the Acoustical Society of America, paper MM10, pp. 377-380, June, 1979.

b) J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-28, pp. 27-34, Feb. 1980.

c) M. Berouti and J. Makhoul, "An Embedded-Code Multirate Speech Transform Coder," IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, pp. 356-359, April 1980.

d) R. Schwartz, J. Klovstad, J. Makhoul and J. Sorensen, "A Preliminary Design of a Phonetic Vocoder Based on a Diphone Model," IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, pp. 32-35, April 1980.

e) J. Makhoul, R. Schwartz, J. Klovstad and J. Sorensen, "Phonetic Recognition and Synthesis in a Total Communication System," Presented at the Dallas Symposium on Voice-Interactive Systems, May 1980.

Copies of papers a) - d) are given in the Appendix.

2. PHONETIC SYNTHESIS

2.1 Introduction

The phonetic synthesizer is one of the two essential components in our proposed very-low-rate (VLR) speech transmission system [2]. Operating at a data rate of about 100 bits per second, the VLR vocoder models speech in terms of phoneme sized units. A block diagram of this system is shown in Figure 1.

This figure shows that input speech undergoes analysis which results in a set of phonemes, phoneme durations and pitches. A phoneme and its associated value of duration and pitch is called a "triplet". Speech rates are typically about 12 phonemes per second, and since each triplet can be encoded into 8 bits, the data rate in the transmission channel is about 100 bits per second. Once the triplets are decoded at the receiving end, a phonetic synthesizer reconstructs the original speech. The phonetic synthesizer must have a stored speech data base to perform this resynthesis. Furthermore, the analysis program must also have access to a data base of phonetically labeled speech. We have designed our phonetic vocoder such that both the analysis and synthesis programs can use essentially the same phonetic data

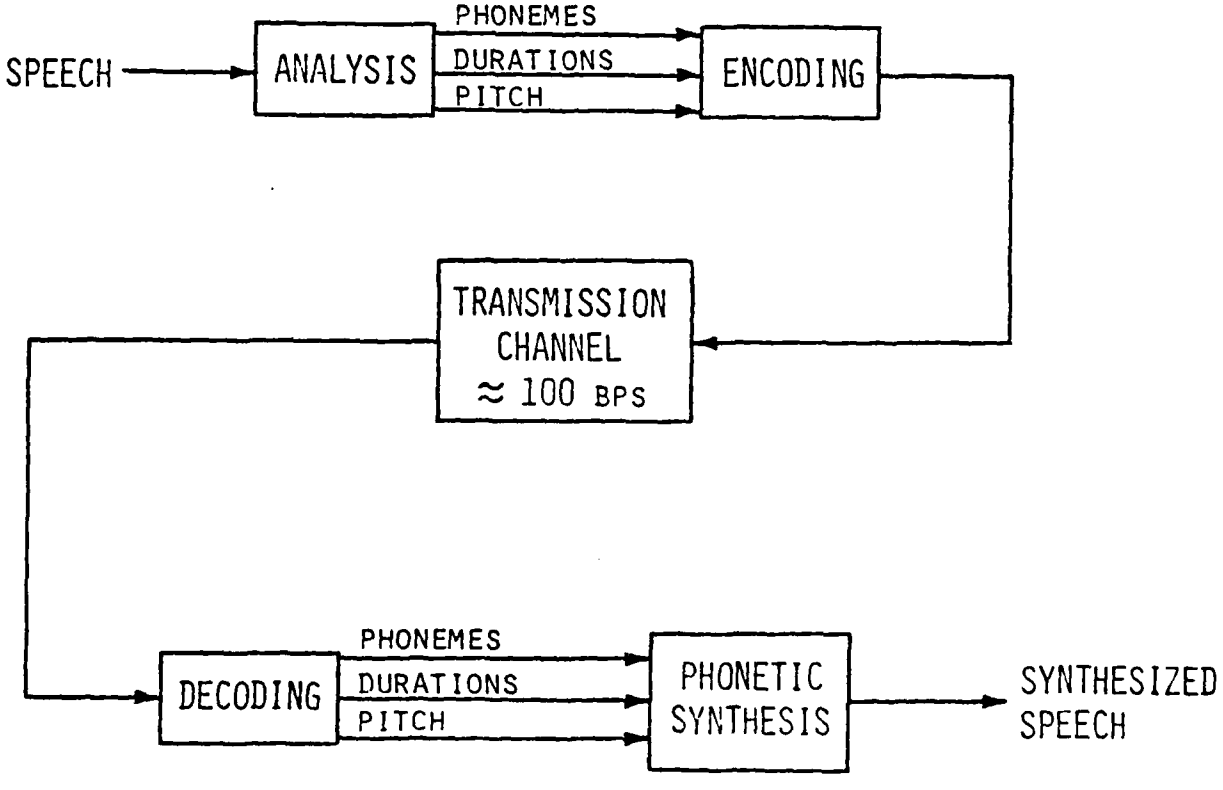


FIG. 1. Block Diagram of the VLR Vocoder

base. Once a suitable data base has been created, the phonetic synthesis program can produce speech. The phonetic synthesis program:

a) translates the input phoneme sequence into a diphone sequence;

b) selects the most appropriate diphone template (depending on the local phonetic context);

c) time-warps each of the diphone templates to produce a gain track and 14 LAR parameter tracks of the specified durations;

d) smooths between adjacent warped diphone templates to minimize gain and spectral discontinuities;

e) reconstructs continuous pitch tracks by linear interpolation of the single pitch values given;

f) determines the cutoff frequency and voicing using knowledge of the phoneme being synthesized;

g) converts resulting LAR parameter tracks to LPC parameter tracks;

h) uses the resulting sequence of LPC, pitch, gain, and cutoff frequency (specified every 10 ms) as input to control an LPC speech synthesizer.

2.2 Diphone Data Base

In this section we briefly review what we mean by a "diphone", and then describe the efforts in labeling the data base and in developing a specification for diphone context.

2.2.1 *The Diphone Concept*

The phonetic synthesizer is designed around the concept of a

"diphone". A diphone is defined as the region from the middle of one phoneme to the middle of an adjacent phoneme. As noted above, the synthesizer must be provided with a data base containing any diphone which would be needed to synthesize a given utterance. For example, synthesizing the word "fan" requires a total of four diphones: [- F], [F AE], [AE N] and [N -] ("-" represents "silence"). The synthesizer would locate templates for these diphones in its data base, perform the appropriate concatenation and time warping of stored spectral and durational parameters, and then synthesize output speech. The consequence of this method is that a fairly large number of diphone templates must be available to the synthesizer as a data base. In our vocoder system, these templates are extracted from nonsense utterances recorded by a single speaker in a quiet room.

2.2.2 Labeling

By labeling, we are referring to the task of marking phonetic boundary locations in the nonsense utterances. For example, consider the nonsense phrase "mee mee meem". This phrase contains several instances of the diphones [M IY] and [IY M], and one instance of the diphones [- M] and [M -]. Using our interactive software and display programs, a phonetician places labels at any or all of the phoneme boundary locations in

this utterance. The speech information for diphones is then appropriately extracted from other files and placed together with all other diphones to form the data base.

A large effort was invested this year in hand-labeling this diphone data base. The initial labeling was completed about half way through the year, and since that time we have added some new diphones as well as eliminated a few unnecessary ones. At present, the data base contains 2733 diphones.

We recently completed the addition of about 50 diphones containing flapped "t" (as in "butter") in various vowel contexts. There are likely to be a few more minor additions to the data base as we encounter diphones in particular phonetic contexts. Appendix A gives a classification of the data base and Appendix B contains an exhaustive list of the 2733 diphones.

2.2.3 Specification of Context

The data base and the phonetic vocoder system are designed so that it is possible to specify explicitly the immediate phonetic context of the diphones. Consider the two phrases "gray train" and "great rain". There is considerable variation in the /t/ in these two cases; in particular, the aspiration following the /t/ release is longer when it is found in the /tr/ consonant

cluster and its burst frequencies are much lower. As a result, we must have two [T R] diphones in the data base; one where the /tr/ is part of a consonant cluster and another where the /t/ and /r/ are separated by a syllable boundary. In the case of the intervening boundary, the diphone is designated as [T#R], and in the consonant cluster as [T R].

The phonetic vocoder also makes use of implicit context. By judicious choice of the inventory of phonemes, we can in some cases eliminate the need to specify explicit context. For example, most phonetic transcriptions of the word "here" contain the phonemes /h/, /i/, and either /r/ or an r-colored schwa vowel. In our vocoder, the transcription is [H] followed by the vowel [IR]. The fact that the /i/ occurs in the context of a following /r/ is therefore taken into account simply by calling /i/ followed by /r/ a separate phoneme, rather than by labeling the /i/ as having occurred in the context of a following /r/. The phonetic inventory also contains four other r-colored vowels (e.g., as in there, far, poor, and four, written [EYR, AR, UR, OR]. "or", respectively).

2.2.4 Rule-Based Synthesis

There are occasions when it is advantageous to synthesize particular phonemes by rule rather than by storing and recalling

parameters of real speech in the data base. One such case is the glottal stop phoneme, as in between the two words "three eagles". The major acoustic manifestation of the glottal stop is a sudden drop followed by a gradual rise in both pitch and energy. We have implemented such a rule in the synthesizer, which eliminates the need to have diphones containing glottal stops in the data base.

Another rule which was recently implemented was to lower the energy during the phoneme "silence" to a low level. We found low level noise existing in our "synthetic silence", and since it was supposed to be silent, the obvious solution was to force it to be so. Both of these rules have been tested and are well received by listeners.

2.3 Diphone Testing

In order to test both the synthesizer and its data base of diphones, we have carried out a process of testing and evaluation. Two basic methods are used. The first is to generate nonsense utterances from a set of rules contained in the synthesis program. The second is to synthesize a complete sentence by inputing a phonetic transcription of an actual sentence, and then comparing the synthetic output to the original

utterance. These methods of testing are described below.

2.3.1 Automatic Sequence Generation

As described in QPR #5 [3], one of the automatic sequence generators in the synthesizer produces nonsense utterances of the form:

$$CVC - VCV - CVCVCV,$$

where C and V are a given vowel and consonant. Once a vowel is specified, the program synthesizes the utterance with all possible consonants in place of C. These utterances are subsequently played back for evaluation by listeners. In this manner, we can test any vowel-consonant or consonant-vowel diphone contained in the data base.

Testing consonant-consonant diphones is accomplished by generating sequences of the form:

$$C_1VC_2\ C_1VC_2.$$

For a given consonant C_1 and vowel V, all possible consonants C_2 are used to generate a group of utterances. The synthesizer also allows a phoneme sequence to be typed in directly from a keyboard, if a specific phoneme string is desired.

These sequence generators and the option for keyboard input allow us to test the diphones in an exhaustive fashion. We have

completed the testing of the consonant-consonant diphones, and have begun testing the consonant-vowel diphones. This testing has allowed us to isolate several program bugs in the diphone concatenation and interpolation routines (See below).

2.3.2 Complete Sentences

The synthesis of complete sentences involves sending the phonetic transcription of a particular sentence (in the form of triplets, each comprising a phoneme, a duration and a pitch value) to the synthesizer, and listening to the resultant speech. About thirty sentences have been synthesized with this method of input.

2.3.3 Debugging

Since exhaustive testing of the diphones began about the middle of the contract year, we have been able to uncover and correct a number of program bugs in the synthesizer. After much searching, we recently found the cause of sudden pops at unvoiced-voiced boundaries, especially in the case of strident fricatives such as "s" or "sh", where there is a large amount of energy in the unvoiced region. Briefly, one of the programs that extracts the waveform parameters that are compiled into the data base was performing incorrectly at the boundary between unvoiced and voiced segments. In QPR #7 [7], we mentioned we had tried to

solve this problem by relabeling the utterances which contain these diphones, but the newer solution is clearly the correct one. A number of errors have also been corrected in the diphone concatenation routines, which have resulted in a more acceptable sounding synthetic speech output.

At the present time the synthetic speech sounds quite natural. As the synthesizer is used in the phonetic vocoder we will occasionally find and solve minor problems with the synthesis to further improve the quality.

2.3.4 Algorithm Refinements

Gain Normalization

As was mentioned in QPR No. 3 [4] and in the last Annual Report [1], each group of recorded diphone utterances has associated with it two normalization utterances. The synthesizer interpolates between the levels in these two normalization utterances to compute a normalization level for each diphone utterance. The synthesizer has been changed to use this normalization level to set the level of each diphone template. That is, the synthesizer amplifies those diphones with a lower normalization level under the assumption that the speaker was talking at a lower overall level for those diphones.

We have mainly been testing the diphones by synthesizing nonsense sentences. However, the complete sentences that we synthesized seemed to have no problems with inappropriate levels similar to the problems we had experienced before.

Time Warping

Two changes were made to the time warping algorithm in the synthesis program. As discussed in the various QPRs, the time-warping algorithm treats the diphone template as being made up of elastic and relatively inelastic regions. For example, the region of the diphone template immediately surrounding the phoneme boundary is not changed by as much as the middle region of the phoneme. The formula that had been used previously resulted in unreasonable time warping when the diphone template was many times longer than the desired phonemes. Consequently, the formula for the time warping factor during the inelastic region has been changed.

Figure 3 shows the formula for the stretch factor during the inelastic region as a function of the stretch factor during the elastic region. The stretch factor is a number that when multiplied by the template duration gives the resulting duration. So if the stretch factor is 2 during the elastic region, that region is stretched to twice its original length. As can be seen

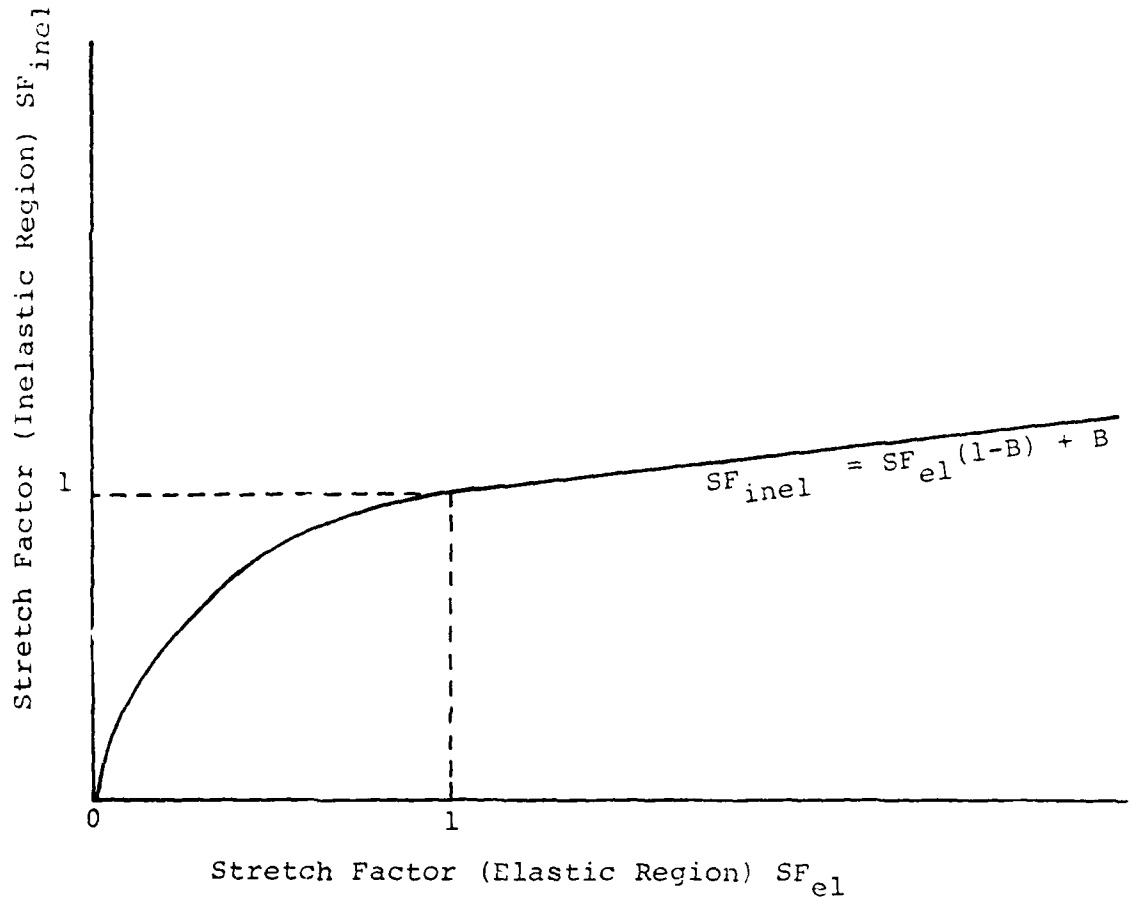


FIG. 2. New Two-Region Formula for Non-Linear Time Warping

in the figure, when the stretch factor in the elastic region is greater than 1 (i.e. the template must be stretched to match the required duration) the inelastic region stretch factor is closer to 1. The formula in this region is

$$SF_{inel} = SF_{el} (1-B) + B$$

where SF_{el} and SF_{inel} are the stretch factors in the elastic and

inelastic regions respectively, and where B is a program variable with a value around 0.9. For example, if $SF_{el}=2$ and B=0.9, then $SF_{inel}=1.1$.

If the stretch factor is less than 1, then the formula is a quadratic:

$$SF_{inel} = -B\ SF_{el}^2 + (1+B)\ SF_{el}$$

This particular quadratic is used because its derivative is the same as that in the previous formula at the boundary ($SF_{el}=1$).

The program is given the durations of the elastic and inelastic regions in each half of the template (each half of the template corresponds to half of one phoneme) and the required total duration for that half of the phoneme in the synthesized speech. It then solves for the stretch factors using the two formulae given above. We have found that this new procedure has resulted in the elimination of some of the unnatural transitions that were present previously.

A second change to the time warping formula involved the effect of speaking rate on the pronunciation of the diphones. We had previously made the assumption that the effect of speaking rate on time-warping of each phoneme was symmetric about the middle of the phoneme. We have observed, however, another

effect. In very slow speech, there tends to be a relatively fast attack for each phoneme and a slower decay. Fig. 4 shows a typical energy track for a vowel spoken at two different rates.

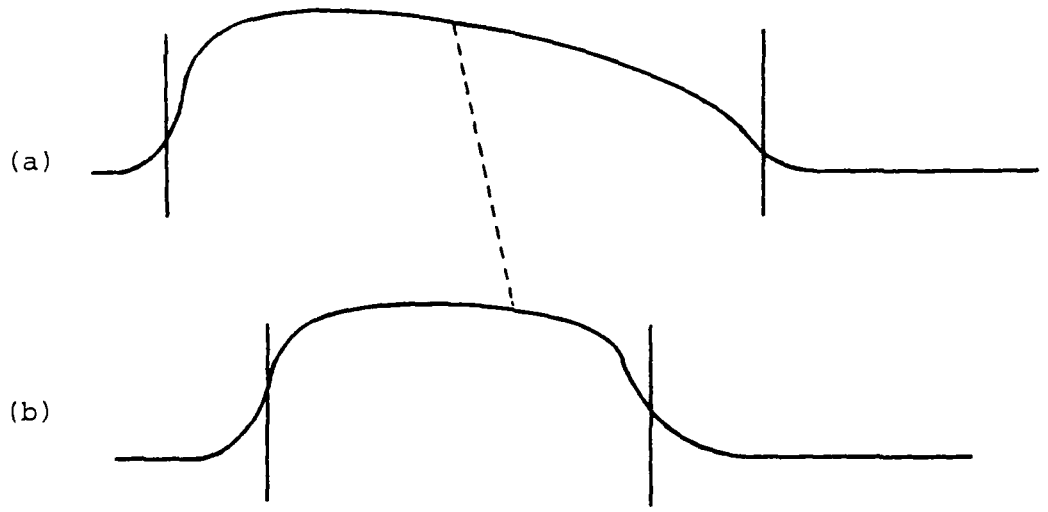


FIG. 3. NON-SYMMETRIC TIME WARPING. a) Shows a typical energy track for two halves of a long vowel phoneme reconstructed from two diphone templates. The dotted line indicates where the templates meet (corresponding to the middles of the phonemes in the nonsense utterances). b) Shows a typical energy track for a shorter vowel phoneme. The middle of the vowel in (a) is mapped (dotted line) to a point after the middle in (b) to produce the desired change in the shape of the contour.

As can be seen, the longer version decays more slowly (even after accounting for the overall duration change) than does the shorter

version. Though not shown here, the spectral parameters also exhibit the same non-symmetric time warping characteristics.

The solution in the time warping algorithm has been to map the middle of a long phoneme (the ends of two diphone templates) to a point after the middle of the required phoneme. Thus, if Fig. 4a shows the energy track as reconstructed by concatenating two diphone templates, then when shortening this phoneme to the desired phoneme duration shown in Fig. 4b, the shape of the contour on the right side is changed by mapping the template onto the phoneme asymmetrically. This mapping (indicated by the dotted line connecting the two energy tracks in Fig. 4) should result in more appropriate pronunciation over a wider range of speaking rates. We have not yet fully tested this new feature.

Transmitted Gain Values

In the original design of the phonetic vocoder, intonation was conveyed by the duration and pitch values. We have found that, occasionally, these are not sufficient; often, the level of energy is inappropriate. The problem is distinct from the gain normalization discussed above, which is intended primarily to adjust for the speaking level at the time of recording.

We have added the capability to transmit with each phoneme (or alternatively, with each vowel) an adjustment to the gain in

the diphone templates. This adjustment is currently a one bit code specifying whether the input phoneme is closer to "normal" loudness, or reduced in level somewhat (by 5 dB). The number of levels and the magnitudes of the adjustments in the program are easily modified. Using a one bit gain adjustment for each vowel would add only 4-5 bits/second to the transmission rate, but is likely to improve the perception of intonation. We have not yet had a chance to measure the improvement in quality due to this addition.

2.4 Text-to-Speech

During this year we interfaced the MITALK unrestricted text-to-speech system to the diphone synthesizer. Input to MITALK is a typed text string, and its final output is a digitized speech waveform. MITALK exists as a number of different programs that are executed one at a time in sequence, with communication between program modules occurring via ordinary text files. Thus it is possible to examine the intermediate output of the system at any point during processing. The last program in the sequence is a synthesis-by-rule module which accepts input in the form of phonemes, durations, pitch, and stress levels. This input is similar to the input required by our diphone synthesizer.

The major task involved was then to convert the MITALK string of phonemes, durations, pitches and stress values into a form compatible with our synthesizer. There are differences in phoneme names, location of phoneme boundaries, scaling of durations and pitches, and specification of certain phonemes in certain contexts. (For details see QPR7).

2.5 Harmonic Deviations

As outlined in the proposal, we started to investigate the feasibility of improving the spectral representation for LPC synthesis by the inclusion of the deviation of each spectral harmonic from the smooth LPC spectral model. In order to test out the principle of harmonic deviations we used it in an analysis-synthesis (vocoder) environment. This work is described in detail in QPR No. 5 [3]

The basic idea used in the Harmonic Deviations Vocoder (HDV) is as follows. The transmitter extracts from the speech signal and sends to the receiver the parameters of a smooth speech spectral envelope model as well as the deviation at each harmonic frequency between the speech spectrum and the spectral envelope model. The receiver generates a pitch-period of the excitation signal from a fixed frequency-dependent harmonic phase spectrum

and the harmonic amplitude spectrum computed from the transmitted deviations. The excitation signal is in turn applied to the filter corresponding to the spectral envelope model to produce the output speech.

As part of our phonetic synthesis project we have developed a floating-point non-real-time simulation of an analysis-synthesis system that uses harmonic deviations. In this simulation, we do not quantize LPC parameters, and we employ the first 15 harmonic deviations. We extract the harmonic deviations from the 10-kHz sampled speech once every 10 ms. Our preliminary experiments have shown that the addition of harmonic deviations to the LPC synthesis significantly increases the naturalness and fullness of the synthesized speech while reducing the buzzy quality.

3. PHONETIC RECOGNITION

As shown in Figure 1, we intend to use the output of a phonetic recognizer to supply the input to the currently available diphone synthesizer. Therefore, the recognizer must extract from an input speech file a sequence of phonemes, each having its own duration and pitch. We have considered two main approaches to recognition: diphone template recognition, and feature-based acoustic-phonetic recognition. The first method compares unknown speech with a large inventory of diphone templates by the use of a distance metric. The program chooses as its answer the sequence of phonemes corresponding to the sequence of diphone templates that matches the input speech most closely according to the distance metric. The second method involves the use of a set of analytical rules that are designed explicitly to make some phonetic distinction. Therefore, these rules can each use specific knowledge pertaining to a specific phonetic distinction - rather than be restricted to a uniform metric. The disadvantage of using acoustic-phonetic rules is that it is hard to arrive at a complete set of rules that result in consistent scores. Also, it has been our experience that mistakes made by such a program tend to be more severe. Therefore, most of our effort during this year has been placed on

the diphone template approach. We still believe, however, that at some future date, acoustic-phonetic rules can be used to disambiguate between particular choices suggested by the first method.

The remainder of this section is organized as follows. Section 3.1 contains an overview of the diphone template recognition method that we use. It details the data structures used and the matching algorithm employed and discusses the scoring philosophy that motivated some of the design decisions. In Section 3.2, we enumerate the implementation issues that have made the speed and storage requirements manageable. Any recognition system requires some training or tuning in order to perform well. The different modes of training that we have implemented and envision for the future are described in Section 3.3. Finally, in Section 3.4, we report on the testing and performance of the initial recognition program, as well as its integration into the phonetic vocoder.

3.1 Diphone Template Recognition

A great deal of thought has gone into the design of our initial phonetic recognition system using diphone templates. What we hope to communicate here is the kinds of issues that were

considered and how their consideration has influenced the design. Also given is the basic structure of each component of the phonetic recognizer as it has evolved over this first year of research.

3.1.1 Motivation for Diphone Template Recognition

The issue discussed here is that of using diphone templates and a matcher to do phonetic recognition. There are two main factors that are relevant to this decision. First, as in the case of phonetic synthesis, using the diphone as a recognition and synthesis unit emphasizes the significance of the phoneme transitions. Accurately modeling the phoneme transitions is important for both recognition and synthesis. Second, we know that the output of this phonetic recognizer is going to be used in conjunction with a phonetic synthesizer that also uses diphone templates. Therefore, there is a strong motivation not only to use diphone templates in the recognizer but also to use the same set of diphone templates. That the same set of diphone templates should be used for both is motivated by the fact that since the recognizer is going to be used in conjunction with the synthesizer, using identical diphone templates for both will, at least, guarantee that the synthesized phonemes are spectrally close to the original.

3.1.2 Diphone Network

The method of phonetic recognition based on the use of a network of diphone templates was described briefly in our proposal [5]. Compiling the diphone templates into a network implicitly forces the matcher to consider only sequences of diphones that are consistent. Since writing the proposal, several of the details have been changed. The diphone network consists of nodes and directed arcs. An example of a simple network is shown in Fig. 5. There are two major types of nodes: phoneme nodes and spectrum nodes. The phoneme nodes (shown as labeled circles) correspond to the midpoints of the phonemes; there is one such node for each phoneme. These phoneme nodes are connected by diphone templates. Each diphone template is represented in the network as a sequence of spectrum nodes (shown as dots).

Each spectrum node consists of a complete spectral template, as well as a probability density of the duration of the node. A spectral template is represented by means and standard deviations for all 14 log-area-ratio (LAR) coefficients and gain. The duration of a node is defined as the number of frames of input aligned with the node. Nodes are connected to each other by directed paths. When two or more consecutive spectra in the

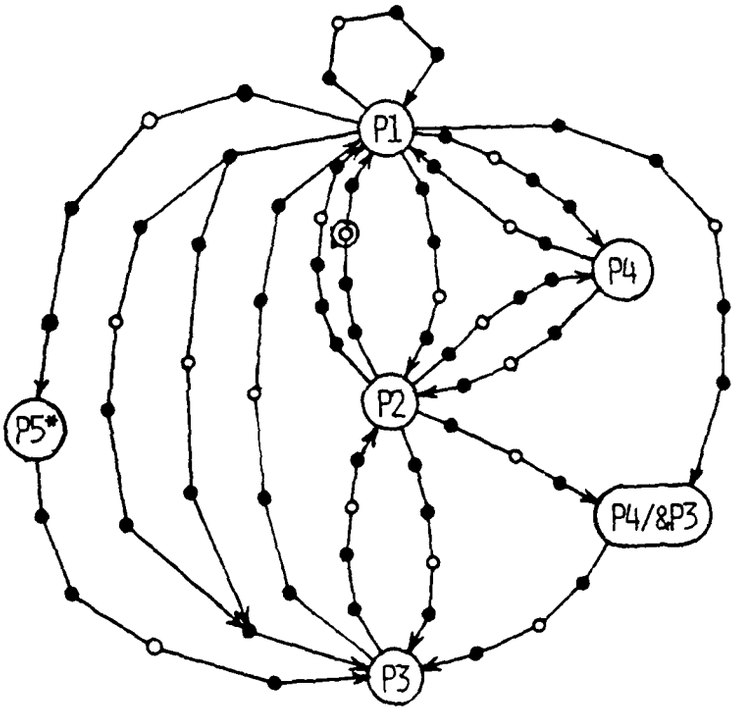


FIG. 4. Example of Diphone Template Network for Five Phonemes

original diphone template are very similar, they are represented by a single spectrum node in the network. Therefore, each spectral node has an implicit self loop. The scoring used in the matcher when this self loop path is taken depends completely on the duration probability density which has been collected during the course of previous training.

The network representation as it has been described so far would easily permit a matcher to determine diphone durations. However, since the synthesizer requires phoneme durations, we explicitly mark each diphone path in the network at the point that corresponds to its phoneme boundary. The open dots in the figure indicate the first spectrum node in the original diphone template that is at or past the labeled phoneme boundary.

Several possible types of network structures are illustrated in the example shown. For example, in Fig. 5 the diphone template P1-P2 is distinct from the template P2-P1. Also note the possibility of diphones of the type P1-P1. The network allows for two or more templates going from one phoneme to another (e.g., P2-P1). Branching and merging of paths within a template is also allowed (e.g., P1-P3). The network allows the specification of diphones in context. The node P4/&P3 represents the phoneme P4 followed only by P3. Thus the template P2-P4/&P3

can be different from the unconditioned template P2-P4. As discussed in Section 2.2.3, we have allowed for the distinction between diphone templates that are within a syllable from those that cross a syllable boundary. In the figure, one of the diphone templates from P2 to P1 is marked as being across a syllable boundary. This syllable boundary is indicated on the spectral node marked as a phoneme boundary (shown as an additional circle around the open dot). Finally, we allow for the representation and compilation of consonant clusters as a single unit - even though the cluster may be several phonemes long. For example, the initial cluster [S-SIT-T-R] as in "string" is acoustically different from the concatenated diphones. Therefore, the sequence is compiled as a complete unit. This means that the intermediate phoneme nodes (-SIT-T-) are used only in this sequence, and they are not shared by other diphones. Thus, there is no branching at these nodes. In the example, the cluster [P1-P5-P3] is shown as a separate path. The phoneme node named P5* indicates one of these unshared intermediate nodes.

Diphone Network Compiler

The program that creates the diphone template network (NETWORK) takes as input a text file similar to that read by the synthesis diphone template compiler (COMPOZ). Like COMPOZ this

compiler also permits the inclusion of incremental changes. That is, we can replace diphone paths, or add additional optional paths to the existing binary file without rerunning the compiler on the original diphone templates. Not only does this preserve program compatibility and eliminate the necessity for redundant representation of the same information but it also greatly reduces the amount of time necessary to produce a new large network if it is only slightly different from a previously compiled one.

Another feature of the format of the information compiled is that statistical information can later be added by the matcher and be available for subsequent incremental additions. Both of the above capabilities were facilitated by the design and use of a memory management package. Thus memory freed by the release of a block of data structure is available for later use. The memory requirements for the network and matcher will be discussed further in Section 3.2.2.

3.1.3 Matcher

Briefly, the analyzer chooses the sequence of templates that best matches the input speech according to a distance measure. Since the received speech is spectrally close to the original, it is hoped that this procedure will suffer minimally from phoneme

recognition errors. The network matcher uses a dynamic programming algorithm which attempts to find the sequence of templates in the network that best matches the input.

The design of the matcher has been strongly motivated by the following 4 considerations.

a) Sound Scoring Strategy

b) Continuous operation

c) Alignment and training capability

d) Efficiency

Of these four considerations the first is perhaps the most important. We feel that the scoring procedure should implement (as accurately as possible) a well formulated scoring strategy. The scoring philosophy requires that the score have components that are derived from a knowledge of the particular path chosen through the network (a priori), the amount of speech aligned with each particular spectral template in the network (durations), and the score derived from a spectral comparison between the input spectrum and the template spectrum. The derivation of the scoring strategy is discussed in detail in QPR #6, Section 3.1 [7].

A second major consideration is that the matcher operate continuously, producing its output as it receives its input -

with some delay - before acquiring all of the input. Thus the matcher can be thought of as producing output as soon as it has what it thinks is sufficient evidence to make a conclusion. Operated in this mode, further input, regardless of what it is, cannot cause the matcher to change previously produced output. This is important because of the intended use of the recognizer in a real-time vocoder application.

The third consideration, that of permitting the user to cause the matcher to find the best alignment for the "correct" phoneme sequence and then use that aligned input is important for the continuing training of the recognition system, as well as to allow debugging of the recognizer. These capabilities will be discussed further in Section 3.3.

The final consideration of efficiency has affected the design of the network and matcher in many ways. This is discussed further in Section 3.2.1.

3.1.4 Search Algorithm

The purpose of the matcher is to find the single path through the network that best matches a sequence of input frames. The program keeps a list of partial "theories" for the best path. A theory consists of a detailed account of how a sequence of

input frames is aligned with the network, along with a total score for that correspondence. For each input frame, the matcher updates each of the theories by the addition of the new frame. This is illustrated in Fig. 6. Each theory spawns at least three new theories: one that corresponds to the new input frame being aligned with the same node as the previous frame, one for each of the nodes immediately following that most recent node, and one for each possible pair of two successive spectral nodes. Thus, in the example shown, the single initial theory (shown by the vertical arrow) would result in 13 new theories - one at each dot. After all theories have been extended, if two or more of the new theories arrive at the same network node on the same input frame, only the best scoring path is kept. Keeping only the best scoring path constitutes our dynamic programming algorithm. The remaining theories are then pruned back so that only the best are kept. There are two parameters that determine how many theories are kept at each step. The first is an absolute maximum number of theories (stack length). The second is the maximum score difference between the best theory and the worst theory kept. If the maximum score difference is set to infinity, then the search is a "bounded breadth search". If the stack length is set to infinity, the algorithm is called a "beam search". The most reasonable tradeoff between speed and accuracy

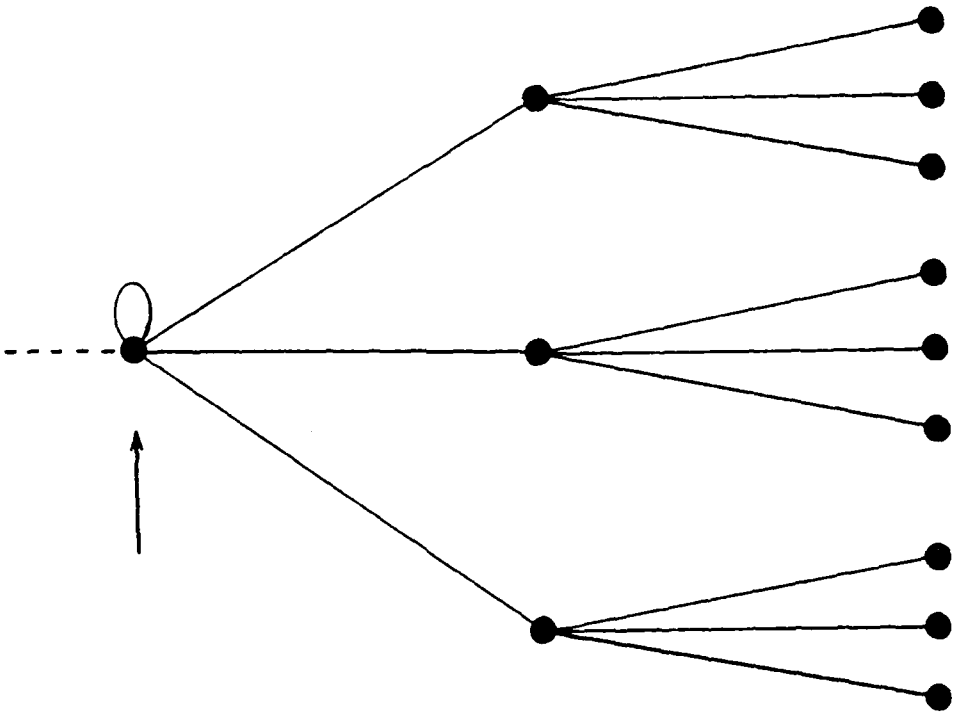


FIG. 5. Theory Update Diagram

is achieved when using both of these methods of pruning. Typically, the number of theories kept is on the order of 1000 per frame.

To decide on the phonemes to transmit, the program examines all the remaining theories after updating each theory with the newest input frame. If all the theories have a common beginning (in terms of exactly the same alignment of the input utterance with the network), the phonemes and durations corresponding to this common beginning are transmitted.

Once a phoneme and its duration have been determined, the pitch is calculated using the weighted least squares method currently used to determine pitch values for our diphone synthesizer.

3.2 Program Efficiency Issues

As mentioned above, efficient operation of the program is an important issue.

3.2.1 Speed

While the simulation is not expected to run in real time, speed is still important. If the program takes several minutes of CPU time to get a result for one sentence, then under a

reasonable computer load average we must wait one half hour to see the result. This reduces the number of variations we can try.

From the start, the program was designed to run as fast as possible. However, the matcher algorithm outlined above requires a large amount of processing. For each 10 ms frame, the program must extend each of about 1000 theories, update them by walking 0, 1 or 2 steps through the network (resulting in about 10,000 theories), score each of these new nodes against the input frame, keep a list of theories sorted by score, keep track of the "genealogy" of all the theories so that it is known when all theories agree. Below, we list some of the search, sorting, and dynamic programming techniques that were employed to reduce the computation by about 4 orders of magnitude from that required for the straightforward implementation.

Theory Merging

Since the program extends all theories by the addition of the same input frame at one time, all theories at all times include the same input. This means that the probabilistic scores on different theories are directly comparable. Therefore, the diphone network was designed such that if two theories merged (i.e. arrived at the same node for the same input frame) only the

best scoring theory is considered. When a theory is advanced, the program checks a word in the new network node to see whether this node has been reached by another theory in this frame. This word points to the previous theory that arrived there. The two theories can be immediately compared (based only on previous scores - without any spectral scoring at this point) and the lower scoring theory deleted. Thus, the program need not create all the data structure and score and sort 10,000 extended theories. Rather it is more like 3000. After all theories have been extended, the program then can score and sort the remaining theories.

Template Scoring

Since an important part of the score is based on the number of input frames that match (are aligned with) a single spectral node, theories that differ in this respect must be kept distinct. This results in the same spectral node being matched against the same input frame several times. For example, one theory may have looped on a particular node two times, and another three times. When these two theories are both advanced by self-looping again, the program detects that this particular score has already been computed. Rather than keeping a large matrix of score as a function of node and input frame pair or, alternatively, clearing the score each time it was used, the program uses a slot in each

spectral node reserved for this purpose. When the spectral distance routine scores this node against an input frame, it records the score and a time stamp in the node. If the spectral distance routine notices that the time stamp in the node matches the current time stamp, then it can merely use the stored score and avoid a distance calculation.

Theory Sorting

As mentioned above, the program needs to know how to keep only the best 1000 or so theories. One way to do this would be to sort the 3000 or so extended theories, and then just keep the best ones. However, since most sort routines require computation proportional to the square of the length of the list, the program uses another method. It uses a partially sorted binary tree of theories. The head of the tree always contains the lowest scoring theory. So a newly extended theory can be compared directly against this theory to decide whether it will be retained. If so, then the new theory ripples down the binary tree until it finds the correct spot. This requires only $Log_2 N$ comparisons. Thus, for N=1000, the savings is $1000/Log_2 1000=100$. Timing statistics have shown us that the time required for this binary partial sort is now small compared to the other processing required in the recognizer.

Spectral Distance Computation

A significant portion of the computation was expected to be in the spectral distance computation. In addition to minimizing the number of times that the routine is called, we would like to make the routine itself as fast as possible. First, the parameters are stored in fixed point format. (We did not lose much accuracy, since the parameters were scaled up before fixing.) Second, a careful examination of the compiled BCPL routine revealed that the routine could be hand coded in assembly language to be 4 times faster.

At this point, timing measurements indicate that less than 10% of the total CPU time is taken by the spectral distance routine.

Beam Search vs. Bounded Breadth Search

While the bounded breadth search described above is efficient, there are times when the difference between the highest score and the lowest score is very large. If we put an upper bound on this difference, then frequently most of the theories can be eliminated. One desirable feature of this "beam search" is that the number of theories kept grows when the decision is not clear cut. Also, it becomes possible to estimate (and control) the probability that the program will accidentally

eliminate a theory that, if kept, would have been the best one.

Future Speed-Up

At this point, the program requires CPU time of about 20-30 times real time for a stack long enough to assure finding a path similar to the best path. Since the time is not spent in the sorting or scoring, it will be hard to make large speed improvements. The bulk of the time required is probably taken up in all the many logical operations used to keep track of theories, walk the network and decide what to do next. In any case, the program is now fast enough so that we can try a reasonable number of variations on a small data base during an overnight batch run. Also, we can run the program comfortably during the day to test and debug new features.

3.2.2 Storage Requirements

In order that the program operate quickly, the entire network must reside in (virtual) memory. If the program had to read pieces of the network from disk files as it ran, it would be several times slower. Therefore, a concerted effort was made to keep the data structures small, and share memory where possible.

The initial diphone template network was compiled from the 2800 or so synthesis diphone templates. The variable-frame-rate compression described in Section 3.1.2 reduced the network by a

factor of 2.5. The spectral parameters were quantized to 9 bit fixed point values to conserve space as well as time. Pointers in the network structure were packed two per word. Even so, this initial network just barely fit in memory (256K words) with the matcher program. We expect that the final network will contain several instances of each diphone. This network could not possibly (even with more compression) fit in one PDP10 address space.

We do not view this as a significant problem for eventual implementation, since large memory chips are rapidly becoming inexpensive. Also, we shall soon be using a DEC VAX 11/780 computer, which has a much larger address space. The personal computers being designed at BBN also feature a large virtual memory.

As a temporary solution to this problem, we have modified the BCPL compiler, our memory management package, and the user programs in order to take advantage of the extended memory capability of the KL10-TOPS20-Release 4 monitor. This currently allows us to use up to 32 full address spaces for our program and data. Since the program uses the BCPL "structure" declarations, switching from half-word pointers to full-word pointers did not require as many source code changes as it would otherwise have

required.

3.3 Training

Each diphone can be spoken in a variety of ways. Therefore, to correctly recognize these many variations as the same phoneme string, the program must be trained with large amounts of natural speech. There are three different ways that we have of training the network.

3.3.1 Manual Training

The most straightforward approach to training the network on a large variety of speech is simply to label (manually transcribe) more speech. This transcribed speech can then be processed by existing programs to add alternate paths to the network.

This method has the advantage of simplicity. There are several disadvantages, however. First, each instance of a particular diphone template would be independent. Therefore, the statistical information (a priori path probabilities, LAR standard deviations, duration probability densities) for each template would be determined from rules rather than training. That is, rather than observing several instances of the same diphone and using the accumulated data to estimate probability

densities, the program would treat each instance as an independent sample, and derive a Gaussian distribution for each parameter using a fixed standard deviation. Second, for those diphones that are common, there might be hundreds of examples before the average number of templates per diphone was high enough. Third, the effort required to accurately label several hundred sentences is tremendous. Fourth, this method requires that the researcher who transcribes the speech know the best way to transcribe each passage (one must often choose between similar labels) and the best rules for placement of phoneme boundaries, as well as be able to use these rules consistently. These problems could significantly affect performance.

Clustering of Diphone Templates

The first and second problems mentioned above can be eliminated by writing a clustering program that would start with the many instances of one diphone, and group them such that there were only a few templates for each diphone, with statistics derived from the data. However, the third and fourth problems (large amount of work, decisions made by humans) would still exist.

3.3.2 Interactive Training

One of the capabilities of the matcher is that the user can

specify what answer it should get for a particular input speech file. The researcher types in a list of phonemes that will be required. If any of the phonemes is in question (e.g. AX vs. AH) then the researcher can just type "ANY" which allows the program to use the closest phoneme at this point. The user can also, if desired, constrain the time that the matcher assigns to the beginning of any phoneme. This is done in a flexible manner, using an interactive command loop. The user constrains the boundary to be either: before t, after t, at t, or between t1 and t2. This "forced alignment" can also be read in from a standard label file, so if the training sentence has already been transcribed, the user need not type in the required phoneme string and times. The user is provided with an editing facility for these lists so that he may insert, delete, or replace items. The user can save alignments on a file and later read them in. There are also global commands that may be used to give the matcher just the right amount of flexibility.

Two likely modes of interaction are outlined below. In the first, the researcher has not carefully transcribed the training utterance. By listening to it briefly, he makes a list of the phonemes (leaving out those about which he is unsure). Then, the matcher is instructed to find the best alignment of the input

utterance to the network under the constraint of the phoneme list. The user compares the printed alignment visually with parameter plots of the input utterance. If the alignment is clearly wrong (a fairly rare occurrence) he constrains the time for the beginning of one or two of the phonemes, and tells the matcher to "do it again".

A second likely mode is that someone has already transcribed the utterance. The researcher reads in the transcription and then, being skeptical, instructs the program to "fuzz" the time constraints by two frames in each direction, so that if the transcriber was off by two frames, the matcher could still find the correct alignment. He then changes any questionable phoneme labels to "ANY" and instructs the matcher to find the best (constrained) alignment.

Once the user is satisfied with part or all of the alignment, he can instruct the matcher to "train" the network using the training utterance. The basic commands available are: Add to Statistics, Add new path, Add new diphone, Save the Network. Each command asks for two time limits specifying a region over which to apply the command. The first command (statistics) causes the program to update the means and standard deviations of the template parameters, as well as the node

duration probability densities. The second command (Add path) causes the program to add in the specified speech as an alternate branch to the section of network against which it was aligned. If the spanned network contains a phoneme boundary, the program asks which frame of the input should be marked as a boundary. The third command (Add diphone) lets the user specify a whole new template for a named diphone.

Using the training commands described, the user actually changes the diphone template network. If he is unsure about or not satisfied with a part of the alignment, he need not use that part for training. At any point in the process, the user can save the updated network on a file.

It is hoped that this procedure, which amounts to interactive clustering, might yield a set of templates that is somehow more self-consistent. One drawback to this approach is that it is a very slow process, requiring the researcher to spend long hours correcting the matcher results.

3.3.3 Automatic Training

One could easily imagine a range of modes of training in which the computer can make more of the decisions. For instance, once the researcher has achieved the desired alignment, he could

issue a command "Train" which would either add to the network statistics or add new paths based on a simple local score threshold. That is, if the match is fairly good, the program would update the statistics. If the match was bad, it would create a new path.

3.4 Testing and Performance

Most of this first year's work on the diphone network recognition program has been spent in designing, implementing, and testing all the features described in the network compiler and matcher. In order to evaluate the performance of the phonetic recognizer, we needed to test it in a reasonable environment. This necessitated integrating the recognizer with the phonetic synthesizer to complete the phonetic vocoder. We also knew that the system would not perform well without being trained. These two efforts are outlined below.

3.4.1 Complete Phonetic Vocoder

To communicate with the phonetic synthesizer, the recognizer must produce a sequence of triplets, each comprising a phoneme, a duration, and a pitch value. As described in Section 3.1.3, each time the matcher drops a theory, it automatically checks whether this now means that there is some part of the answer that is

common to all theories. This may happen one frame at a time, or sometimes deletion of one theory may eliminate the last conflict for several phonemes worth. In any case, for each frame of the input, the program types out (to a terminal or to a disk file) a symbol indicating the type of alignment: self loop, advance to a new node, share the input frame between two nodes, whether the node is marked as a phoneme or syllable boundary. The program can also output the various components of the score for each input frame, as well as the cumulative scores. Also, whenever the matcher detects that the best path crosses a phoneme node in the network, it types out the name of the phoneme node.

An addition was made to these typeout routines, such that they would remember the names of all the phoneme nodes crossed, the time of the phoneme boundaries, and whether the boundary was also a syllable boundary. The program can then write a standard transcription file which can be read in by the phonetic synthesizer (as well as many other utility programs). The synthesizer then reads in the pitch file for the input utterance, and models the pitch by a weighted piecewise linear fit to determine the pitch values. In a real phonetic vocoder, this last function would reside in the recognizer.

Below we describe some of the benchmark experiments

performed.

3.4.2 Benchmark Experiments

The first, very short experiment performed, was to try to phonetically vocode a sentence. We used the untrained network constructed from only the synthesis diphone templates. 66% of the phonemes were correctly recognized. However, there were many extra phonemes in the output string. Upon examination of the sequence of phonemes, we felt that it would be unintelligible when synthesized. However, when we used this output in the synthesizer, we found the sentence somewhat recognizable, though there were problems. On playing the sentence to several naive listeners, they understood most or all of the sentence. This led us to believe that we were correct in assuming that even when the phonemes were wrong, the spectrum would be close enough.

At this point, we wanted to measure the effect that training would have. However, we did not want to wait until the network was fully trained before testing it. Therefore, we recorded the first paragraph of the Rainbow Passage two times (about 30 sec or 320 phones of speech each). The first reading of the paragraph was manually transcribed and used to augment the network (described in Section 3.3.1). We then tried to recognize the second reading of the paragraph. From careful examination of the

second reading, we felt that 5% of the phonemes were actually different, i.e., they were pronounced differently. This meant that about 90% of the diphones in the second reading of the paragraph had at least one template in the network that came from natural speech - as opposed to the 2800 diphone templates extracted from nonsense utterances that made up the rest of the network.

The result of this small test was that 79% of the phonemes in the second reading were recognized correctly. There were still some extra phonemes in the output. On playing the synthesized speech resulting from this output, most listeners understood most of the sentences. However, we felt that both the quality and intelligibility would need to be improved before this system would be acceptable.

The above result is encouraging. We must remember, however, that the experiment was optimistic in that the diphone templates were extracted from the same global environment as they would be used. We might hope that the fact that in the final system, all diphones will have several training samples, will make up for not always having diphones trained in the same context. The mode of training used in this experiment did not permit the use of statistics. Also, we would hope that when the other diphones in

the network have been trained, they would be less likely to be confused with the correct diphones.

In a third, short experiment, we carefully constructed sentences that contained a mixture of "trained" and "untrained" diphones. One half of the diphones had not been trained on natural speech. Those diphones that had been trained were used in a completely different environment from that of the training sentences. When tested on the matcher, 78% of the trained diphones were recognized correctly, while only 40% of the untrained diphones were recognized. Again, this test was not extensive enough to predict the final performance of the system after extensive training.

The above experiments tell us that training of the network is very important. It also tells us that there is a significant difference between using natural speech versus nonsense speech. Therefore, we may decide to delete each nonsense diphone template from the network as soon as there is a corresponding template from natural speech to replace it. This capability has not yet been added to the network compiler or matcher, though it would probably be relatively straightforward.

3.5 Conclusion

We have, during the past year, designed and implemented an initial version of a phonetic recognition program particularly suited to very-low-rate phonetic vocoding. The program uses a network of diphone templates for recognition, which makes it most compatible for use with the diphone template synthesizer. The program was also designed to be flexible and allow interactive training, so as to be useful as a research tool.

Our preliminary results, based on a small amount of training to the speech of a single speaker are encouraging. While the system will clearly need more training and some logical modifications, we feel that the final outcome will be good.

4. MULTIRATE CODING

4.1 Introduction

The goal of the speech compression project this year has been to design and test an embedded-code multirate adaptive transform coder. The objective is to have a system capable of operating at an arbitrary data rate in the range 2.5 to 9.6 kb/s. It was also desired to let the transmission channel vary the bit rate while the algorithm at the transmitter remained unaffected. This last constraint is satisfied by embedding the codes, i.e., arranging the codes in such a manner that, when some bits are discarded by the channel, the remainder of the received bits can still be decoded in a meaningful way to produce a speech output.

To meet the requirements of the project, we chose a frequency-domain approach or adaptive transform coding (ATC), combined with our newly developed methods of high frequency regeneration (HFR) by spectral duplication. The combined system is basically a linear prediction (LPC) vocoder operating at 2.5 kb/s and embedded in a voice-excited coder which provides higher speech quality at higher data rates. The voice-excited coder, realized in the frequency domain by ATC techniques, always operates at a fixed bit rate, say 16 kb/s. The multirate feature

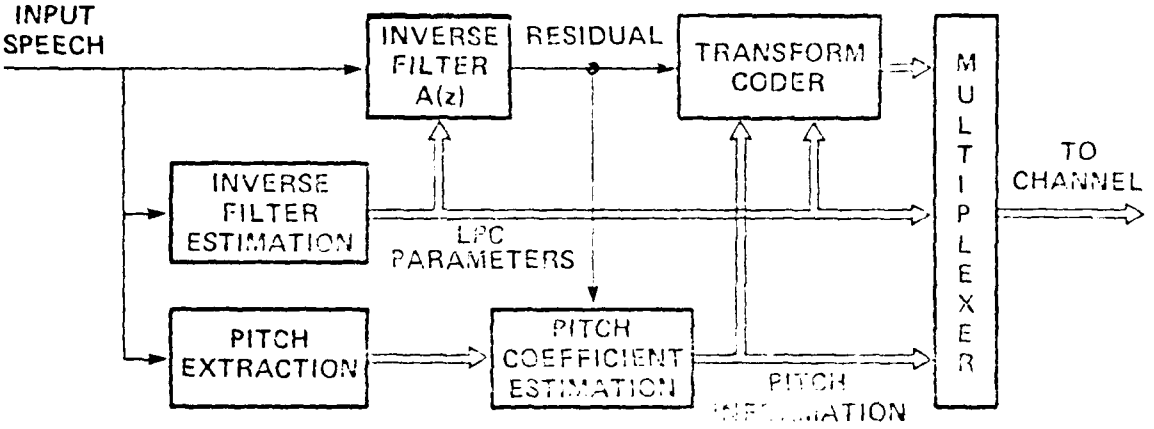
of the system is realized by allowing the channel to strip off some of the bits from the high rate system thereby achieving lower data rates. Such a hybrid LPC-ATC system offered great flexibility in meeting the goals of the project.

In the remainder of this section we shall describe the system briefly and highlight the various aspects of our work.

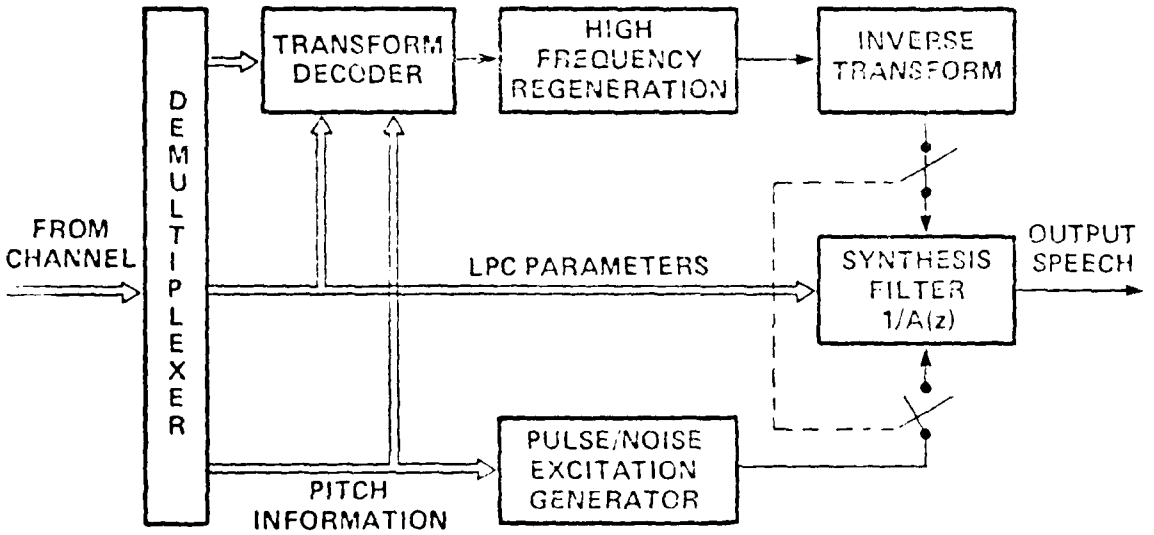
4.2 System Description

The complete multirate coding system is shown in Fig. 7. At the transmitter, the input speech signal is analyzed as in the usual LPC vocoder, namely, LPC parameters, pitch, and gain are derived, quantized, and coded. In addition, the speech signal is inverse filtered, and the discrete cosine transform (DCT) of the residual is taken. With pitch known, the coefficient of a one-tap pitch filter is derived from the residual. This pitch filter, together with the LPC parameters, form the spectral model to be used in the bit allocation process to perform the coding of the DCT. The coded DCT components are then delivered to the channel which may transmit all of the bits or suppress some of them.

At the receiver, the DCT codes are decoded to form a baseband DCT. The fullband DCT of the residual is then



TRANSMITTER



RECEIVER

FIG. 6. Block Diagram of a Multirate Speech Transform Coder

reconstructed using the method of HFR by spectral duplication. An inverse DCT yields the time-domain reconstructed residual waveform to be used as input to the all-pole LPC synthesis filter to produce the speech output. In case all of the DCT components are suppressed by the channel (lowest data rate), the receiver becomes identical to that of a pitch-excited LPC vocoder and uses a pulse/noise source as excitation to the synthesis filter.

4.3 Summary of Work Done

4.3.1 Modified ATC

Traditionally, in conventional ATC, one codes the DCT of the speech signal itself. Perhaps the most salient feature in our implementation of ATC is that we code the DCT of the linear prediction residual. In the proposal [5] for this work and in the sixth quarterly progress report (QPR #6) [7] on this contract, we explained how one can quantize the DCT of the residual rather than the DCT of speech. We explained that not only both approaches yield the same signal to quantization noise ratio, but also some advantages are accrued when coding the DCT of the residual. One such advantage is that, when transmitting the DCT of the residual, the all-pole synthesis required at the receiver helps smooth frame-boundary discontinuities that would normally be present because of frequency-domain quantization.

Another advantage is that the DCT of the residual has a flat spectral envelope. This property is particularly desirable for regenerating the missing high-frequency components by spectral duplication of the baseband. The HFR process creates a fullband spectrum that contains the pitch information and has a flat spectral envelope. Thus, the reconstructed spectrum is particularly well suited for all-pole synthesis. Our approach is to be contrasted with the case where the DCT of speech is transmitted. Such a case is the frequency-domain counterpart of so-called voice-excited coders that transmit a baseband of the speech signal. Time-domain implementations of voice-excited and residual-excited coders have shown that the quality of the output speech obtained with residual-excited coders is always preferred to that obtained with voice-excited coders.

4.3.2 Bit Allocation

The spectral model of speech used in bit allocation was discussed in QPR #5 [3]. Briefly, it consists of two components: a smooth (LPC) spectral envelope, and a model for the harmonic structure of the spectrum (the pitch filter). In QPR #6 [7] we showed how bit allocation can be interpreted as uniform quantization of the logarithm of the weighted spectral model. The weighting applied to the spectral model is the reciprocal of

the desired spectral envelope for the output noise. The process of uniform quantization of the log-spectral-model is shown in Fig. 8. The dashed curves in the figure define the quantization intervals and they take on the shape of the desired spectral envelope of the output noise. In the absence of noise shaping, these curves would be straight horizontal lines. In our implementation of bit allocation, the spacing between the curves is 5 dB instead of the traditionally used 6 dB. In fact, ideally, the spacing between the curves should not be the same everywhere, i.e., the quantization should be non-uniform. This subject was discussed in detail in QPR #6 [7].

4.3.3 Embedded Multirate Coding

For each input frame, the transmitter transmits a block of bits which is divided into two major parts. The first part contains the bits representing the system parameters and the second part contains the bits representing the DCT components. One such block of bits is shown in Fig. 9. The first part of the parameter codes is essentially identical to what is transmitted by an LPC vocoder. The additional parameter codes needed are: 1 pitch tap for the harmonic model of the DCT spectrum, and HFR codes to be discussed in Section 4.3.4. The maximum data rate of 16,000 b/s is achieved when the DCT of the fullband residual is

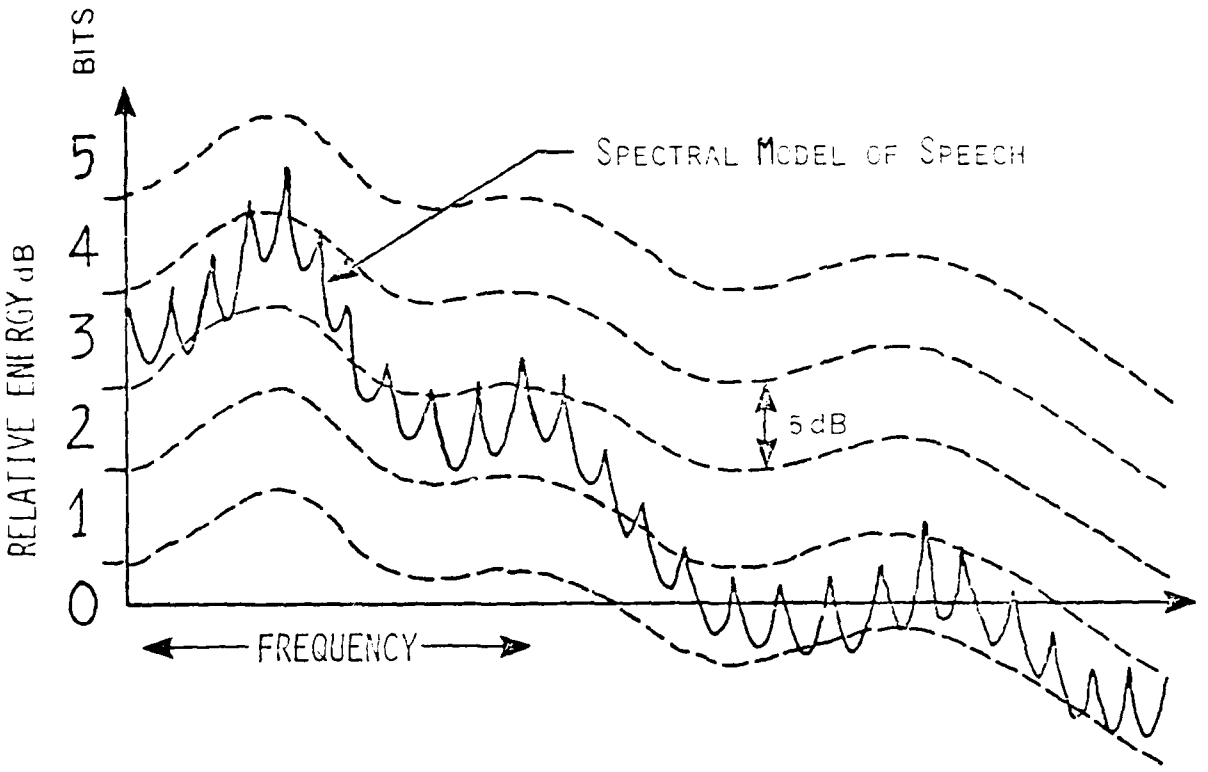


FIG. 7. Bit Allocation in ATC by Uniform Quantization of the Log-Spectral-Model

transmitted. The minimum data rate achievable by the system takes place when all the codes representing the DCT components are suppressed by the channel. Intermediate data rates are achieved by stripping off bits from the high data rate system. Stripping off bits results in the suppression of low-variance frequency components and/or high-frequency components, depending

on how the stripping is performed. This aspect of the system is explained below.

It is worth mentioning here that the transmitter itself can strip off bits prior to transmission in the same manner as is done by the channel. Thus, when a system is first turned on, the initial bit rate need not be high and traffic congestion can be avoided. Note that the receiver does not need to know where in the system the bits are discarded.

The codes representing the DCT components are arranged in a certain order prior to transmission. This ordering determines which bits get discarded first, which, in turn, determines whether high frequency components or low variance components get discarded. To study the tradeoff between the number of transmitted bits, the quantization accuracy, and the number of transmitted frequency components, we investigated three bit-ordering techniques. These were explained in detail in QPR #7 [6] and we summarize them below.

The first bit-ordering technique, which we call the baseband coder approach, is the simplest: the codes are arranged by order of increasing frequency. When the channel strips off bits from the end of each block, the high frequency components are

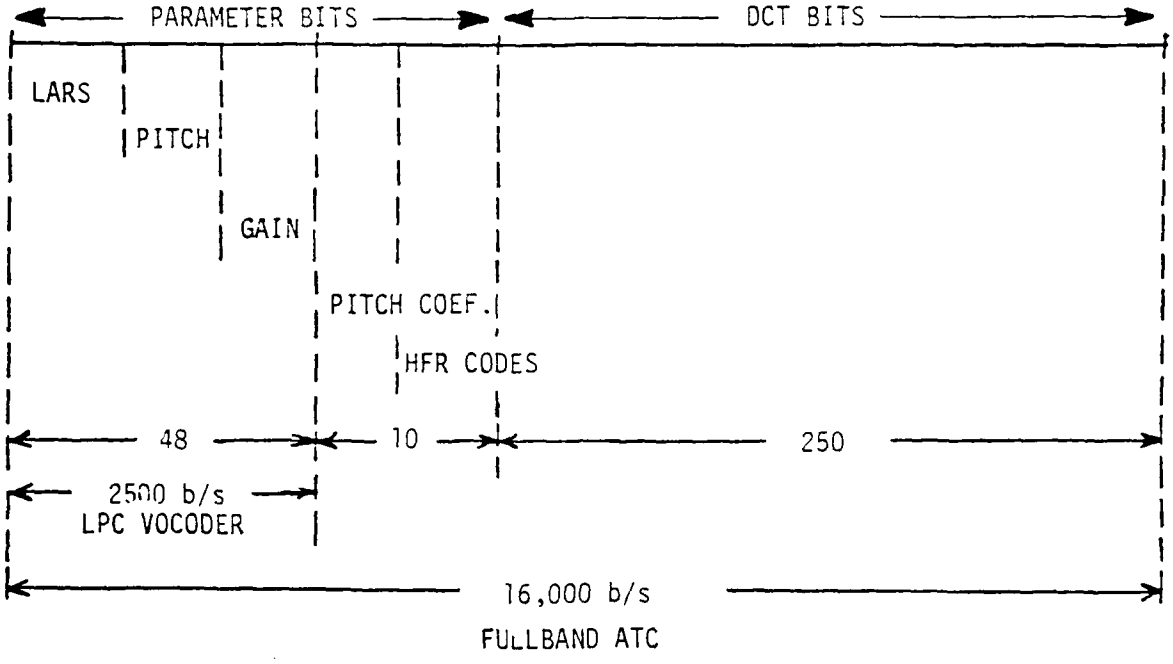


FIG. 8. One Block of Bits Per Frame Illustrating the Embedded-Code Multirate Feature of the System

discarded first. The remaining codes represent a low frequency portion of the total bandwidth referred to as a baseband. As in a baseband coder, the receiver regenerates the missing high-frequency components. We use the method of high-frequency regeneration (HFR) by spectral duplication, which is explained in Section 4.3.4.

In the second bit-ordering technique we discard the least significant bit of each DCT code, starting with the high frequency components, until the desired rate is reached. If need be, the next-to-least significant bits of each code are also discarded. This method decreases the quantization accuracy uniformly over the whole band. It is equivalent to having performed the initial bit allocation with fewer bits, e.g., 120 bits instead of 250 bits for a rate of 9 kb/s instead of 16 kb/s.

In the third ordering technique, the discarded bits represent DCT components with a relatively low variance. Since the codelength obtained by bit allocation is directly proportional to the relative variance of the DCT components, this technique is realized by discarding all 1-bit codes, then all 2-bit codes, etc..., until the desired rate is achieved. Referring back to Fig. 8, we can see that this technique corresponds to having merged together into one large level, two or more inner levels of the uniform quantization of the log-spectral-model. For example, at 9.6 kb/s, the method corresponds roughly to having merged the 0-bit and 1-bit steps into one large 0-bit step.

When comparison is made at the same bit rate, the second and third ordering techniques yield a baseband that is generally

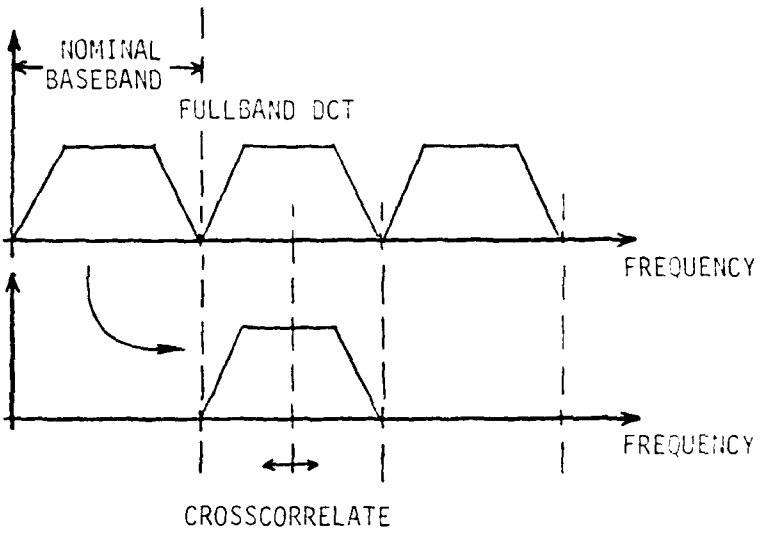
narrower than that of the baseband coder approach. In exchange for a smaller baseband width, the second and third techniques provide some additional DCT components scattered in the high-frequency region. Further details on embedded coding are found in QPR #7 [6], where we concluded that the first bit-ordering technique, i.e., the baseband coder approach, yields the best output speech quality for data rates in the range 6.4 to 9.6 kb/s.

4.3.4 High Frequency Regeneration

The aim of high frequency regeneration (HFR) is to recreate the missing high frequency components of a signal. Traditionally, some form of non-linear distortion of the time-domain signal is used, e.g., waveform rectification. More recently, we introduced HFR methods where the missing components are regenerated by duplicating the baseband components at high frequencies [8]. We call this approach the spectral duplication method of HFR. Spectral duplication aims at reconstructing a fullband spectrum that has a harmonic structure and a flat spectral envelope. Care must be taken not to interrupt the harmonic structure of the signal spectrum. Our initial efforts were described in a previous annual report [1] while our more recent work is detailed in QPR #7 [6]. We now summarize the final algorithm that we are currently using.

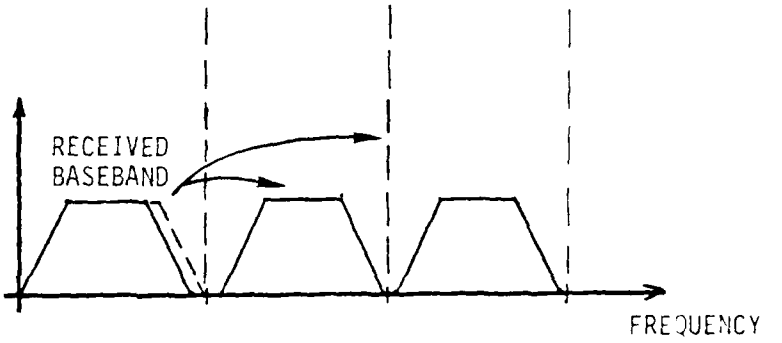
The HFR method is illustrated in Fig. 10. The analysis process needed at the transmitter is as follows. First, a nominal baseband of fixed width (1 kHz) is translated up in frequency to fill the region from 1 to 2 kHz. Second, to find an optimal position for the baseband, the baseband is cross-correlated with the actual fullband DCT present in that region. The lag at which the correlation is maximum is interpreted to be the point where the baseband best duplicates or best matches the original DCT components. It is the harmonic structure of the DCT that helps lock the baseband into place. Thus, the method preserves the harmonic continuity of the DCT. The lag value is transmitted by means of a 3-bit HFR code, which accommodates lags between -3 and +4 spectral points. The same process is repeated for higher frequency bands.

At the receiver, the received baseband is translated to its nominal position (1 to 2 kHz) and additionally shifted by a few spectral points as indicated by the HFR code. Since the received baseband is seldom equal to 1 kHz in width, the regenerated bands may either overlap one another or have gaps between them. These cases are easily taken care of (QPR #7 [6]). We now report on the latest results obtained with the most recent versions of the various aspects of the multirate system.



A. At the transmitter

1. Choose point of maximum correlation.
2. Transmit 3-bit HFR codes



B. At the receiver

1. Translate received baseband
2. Fill gaps

FIG. 9. High Frequency Regeneration by Spectral Duplication

4.3.5 Results

With the HFR method as outlined above, the results for the baseband coder approach of embedded multirate coding are shown in Table 1.

TOTAL RATE kb/s	AVERAGE RECEIVED BASEBAND WIDTH Hz	AVERAGE CODELENGTH bits/sample
16.0	3333	1.95
9.6	1400	2.3
7.2	870	2.4
6.4	670	2.5

TABLE 1. Results Obtained with the Baseband Coder Approach for Multirate Coding

From the table it can be seen that at 6.4 kb/s the average width of the received baseband is about 670 Hz. This is quite a narrow baseband and constitutes the major obstacle for lowering the data rate any further. At 6.4 kb/s, the quality of the synthetic speech is acceptable, but lowering the bit rate further causes the speech to be quite rough, hollow-sounding, and with occasional pops. Thus, for rates below 6.4 kb/s, we recommend

using the 2.5 kb/s LPC vocoder. For rates larger than 6.4 kb/s, the average baseband width is substantially increased and the quality of the speech improves markedly. The quality at 16 kb/s is very close to the original.

4.4 Conclusions

In this project, we have capitalized on the advantages of a frequency-domain approach to realize a versatile embedded-code multirate speech coding system. Some of the advantages of our system are: the ease with which spectral noise-shaping can be implemented, the ease with which the tradeoff between quantization accuracy and baseband width can be studied, the ease with which the codes can be embedded for multirate operation, and the ease with which high frequency regeneration can be done for effective voice excitation. In addition, the system is suitable for real-time implementation on existing array processors.

5. REFERENCES

[1] M.Berouti, L. Cosell, J. Klovstad, J. Makhoul, R. Schwartz and J. Wolf, "Speech Compression and Synthesis," Annual Report, Contract No. F19628-78-C-0136, BBN Report No. 4159, Aug. 1979.

[2] J. Makhoul, C. Cook, R. Schwartz and D. Klatt, "A Feasibility Study of Very Low Rate Speech Compression Systems," Report No. 3508, Bolt Beranek and Newman, Inc. Cambridge, MA, Feb. 1977.

[3] M. Berouti, J. Klovstad, J. Makhoul, R. Schwartz and J. Sorensen, "Speech Compression and Synthesis," Quarterly Progress Report No. 5, BBN Report No. 4266, 8 June 1979 to 7 Sept. 1979.

[4] L. Cosell, A. Huggins, J. Klovstad, J. Makhoul, R. Schwartz and J. Wolf, "Speech Compression and Synthesis," Quarterly Progress Report No. 3, BBN Report No. 4061, 6 Oct. 1978 to 5 Jan. 1979.

[5] J. Makhoul, et al., "Proposal for Continuation of Research on Narrowband Communication," BBN Proposal No. P79-ISD-20, Nov. 1978.

[6] M. Berouti, J. Klovstad, J. Makhoul, R. Schwartz and J. Sorensen, "Speech Compression and Synthesis," Quarterly Progress Report No. 7, BBN Report No. 4354, 8 Dec. 1979 to 7 March 1980.

[7] M. Berouti, J. Klovstad, J. Makhoul, R. Schwartz, J. Sorensen, "Speech Compression and Synthesis," Quarterly Progress Report No. 6, BBN Report No. 4277, 8 Sept. to 7 Dec. 1979.

[8] J. Makhoul, and M. Berouti, "High-Frequency Regeneration in Speech Coding Systems," Int. Conf. on Acoustics, Speech and Signal Processing, Washington, DC, pp. 428-431, April 1979.

APPENDIX A

CATEGORIZATION OF DIPHONES

APPENDIX A
CATEGORIZATION OF DIPHONES

C_1V_1 DIPHONES (673)

C_1 P T K B D G CH JH F V TH DH (25)
S Z SH ZH W Y R L M N HH NX DX

V_1 IY IH EY EH AE AA AO AH OW UH UW ER AY1 (27)
OY1 AW YU IR OR AR EYR UR AX IX AXR EL EM EN

Additions: [TQ EL] [TQ EM] [TQ EN]
Exceptions: [W YU] [Y YU] [R YU] [DX YU] [DX UR]

V_2C_2 DIPHONES (662)

V_2 V_1 except [AY1] is replaced by [AY2] and [OY1] is replaced by [OY2] (27)

C_2 SIP SIT SIK SIB SID SIG SIC SIJ F V TH DH (25)
S Z SH ZH W Y R L M N HH NX DX

Exceptions: [YU W] [YU Y] [YU R] [AX DX] [IX DX] [AXR DX] [YU DX]
[EL NX] [EM NX] [EN NX] [EL DX] [EM DX] [EN DX]

V_3V_4 DIPHONES (290)

V_3 AA AO AW AY2 IY ER EY OW OY2 UW EL (11)

V_4 V_1 less [EM] [EN] (25)

Additions: [IH EL] [EH EL] [AE EL] [AH EL] [UH EL] [YU EL] [EL EM]
[IR EL] [OR EL] [AR EL] [UR EL] [AX EL] [IX EL] [EL EN]
[AXR EL] [EYR EL]
Exceptions: [EL EL]

$C_3\#C_4$ DIPHONES (462)

C_3 P T K B D G CH JH F V TH DH (20)
S Z SH ZH R M N NX

C_4 C_2 less [NX] [DX] (23)

Additions: [N DX] [HH W]

Stop Consonant Diphones with Context (400)

SIX X / & V_1 V_1 defined above (216)
SIX X / & # C_4 C_4 defined above (184)

X P T K B D G CH JH

Dipthongs with Context (100)

Y1 Y2 / & V_4 V_4 defined above (50)
Y1 Y2 / & C_2 C_2 defined above (50)

Y AY OY

Diphones with Silence (88)

- V_5 V_5 V_1 less [AX] [IX] [AXR] [EL] [EM] [EN] (21)
- C_4 C_4 defined above (23)
V_6 - V_6 V_3 plus [EM] [EN] (13)
C_3 - C_3 defined above (20)
SIX X / & - X defined above (8)
Y1 Y2 / & - Y defined above (2)
- - (1)

Diphones from Clusters (58)

Grand Total (2733

APPENDIX B

EXHAUSTIVE LIST OF 2733 DIPHONES

APPENDIX B
EXHAUSTIVE LIST OF 2733 DIPHONES

- -	- AA	- AE	- AH	- AO	- AR
- AW	- AY1	- DH	- EH	- ER	- EY
- EYR	- F	- HH	- IH	- IR	- IY
- L	- M	- N	- OR	- OW	- OY1
- R	- S	- SH	- SIB	- SIC	- SID
- SIG	- SIJ	- SIK	- SIP	- SIT	- TH
- UH	- UR	- UW	- V	- W	- Y
- YU	- Z	- ZH	AA -	AA AA	AA AE
AA AH	AA AO	AA AR	AA AW	AA AX	AA AXR
AA AY1	AA DH	AA DX	AA EH	AA EL	AA ER
AA EY	AA EYR	AA F	AA HH	AA IH	AA IR
AA IX	AA IY	AA L	AA M	AA N	AA NX
AA OR	AA OW	AA OY1	AA R	AA S	AA SH
AA SIB	AA SIC	AA SID	AA SIG	AA SIJ	AA SIK
AA SIP	AA SIT	AA TH	AA UH	AA UR	AA UW
AA V	AA W	AA Y	AA YU	AA Z	AA ZH
AE DH	AE DX	AE EL	AE F	AE HH	AE L
AE M	AE N	AE NX	AE R	AE S	AE SH
AE SIB	AE SIC	AE SID	AE SIG	AE SIJ	AE SIK
AE SIP	AE SIT	AE TH	AE V	AE W	AE Y
AE Z	AE ZH	AH DH	AH DX	AH EL	AH F
AH HH	AH L	AH M	AH N	AH NX	AH R
AH S	AH SH	AH SIB	AH SIC	AH SID	AH SIG
AH SIJ	AH SIK	AH SIP	AH SIT	AH TH	AH V
AH W	AH Y	AH Z	AH ZH	AO -	AO AA
AO AE	AO AH	AO AO	AO AR	AO AW	AO AX
AO AXR	AO AY1	AO DH	AO DX	AO EH	AO EL
AO ER	AO EY	AO EYR	AO F	AO HH	AO IH
AO IR	AO IX	AO IY	AO L	AO M	AO N
AO NX	AO OR	AO OW	AO OY1	AO R	AO S
AO SH	AO SIB	AO SIC	AO SID	AO SIG	AO SIJ
AO SIK	AO SIP	AO SIT	AO TH	AO UH	AO UR
AO UW	AO V	AO W	AO Y	AO YU	AO Z
AO ZH	AR DH	AR DX	AR EL	AR F	AR HH
AR L	AR M	AR N	AR NX	AR R	AR S
AR SH	AR SIB	AR SIC	AR SID	AR SIG	AR SIJ
AR SIK	AR SIP	AR SIT	AR TH	AR V	AR W
AR Y	AR Z	AR ZH	AW -	AW AA	AW AE
AW AH	AW AO	AW AR	AW AW	AW AX	AW AXR
AW AY1	AW DH	AW DX	AW EH	AW EL	AW ER
AW EY	AW EYR	AW F	AW HH	AW IH	AW IR

AW IX
AW IY
AW L
AW M
AW N
AW NX
AW OR
AW OW
AW OY1
AW R
AW S
AW SH
AW SIB
AW SIC
AW SID
AW SIG
AW SIJ
AW SIK
AW SIP
AW SIT
AW TH
AW UH
AW UR
AW UW
AW V
AW W
AW Y
AW YU
AW Z
AW ZH
AX DH
AX EL
AX F
AX HH
AX L
AX M
AX N
AX NX
AX R
AX S
AX SH
AX SIB
AX SIC
AX SID
AX SIG
AX SIJ
AX SIK
AX SIP
AX SIT
AX TH
AX V
AX W
AX Y
AX Z
AX ZH
AXR DH
AXR EL
AXR F
AXR HH
AXR L
AXR M
AXR N
AXR NX
AXR R
AXR S
AXR SH
AXR SIB
AXR SIC
AXR SID
AXR SIG
AXR SIJ
AXR SIK
AXR SIP
AXR SIT
AXR TH
AXR V
AXR W
AXR Y
AXR Z
AXR ZH

AY1 AY2 / & -
AY1 AY2 / & AA
AY1 AY2 / & AE
AY1 AY2 / & AH
AY1 AY2 / & AO
AY1 AY2 / & AR
AY1 AY2 / & AW
AY1 AY2 / & AX
AY1 AY2 / & AXR
AY1 AY2 / & AY1
AY1 AY2 / & DH
AY1 AY2 / & DX
AY1 AY2 / & EH
AY1 AY2 / & EL
AY1 AY2 / & ER
AY1 AY2 / & EY
AY1 AY2 / & EYR
AY1 AY2 / & F
AY1 AY2 / & HH
AY1 AY2 / & IH
AY1 AY2 / & IR
AY1 AY2 / & IX
AY1 AY2 / & IY
AY1 AY2 / & L
AY1 AY2 / & M
AY1 AY2 / & N
AY1 AY2 / & NX
AY1 AY2 / & OR
AY1 AY2 / & OW
AY1 AY2 / & OY1
AY1 AY2 / & R
AY1 AY2 / & S
AY1 AY2 / & SH
AY1 AY2 / & SIB
AY1 AY2 / & SIC
AY1 AY2 / & SID
AY1 AY2 / & SIG
AY1 AY2 / & SIJ
AY1 AY2 / & SIK
AY1 AY2 / & SIP
AY1 AY2 / & SIT
AY1 AY2 / & TH
AY1 AY2 / & UH
AY1 AY2 / & UR
AY1 AY2 / & UW
AY1 AY2 / & V
AY1 AY2 / & W
AY1 AY2 / & Y
AY1 AY2 / & YU
AY1 AY2 / & Z
AY1 AY2 / & ZH
AY2 - / AY1 &
AY2 AA / AY1 &
AY2 AE / AY1 &
AY2 AH / AY1 &
AY2 AO / AY1 &
AY2 AR / AY1 &
AY2 AW / AY1 &
AY2 AX / AY1 &
AY2 AXR / AY1 &
AY2 AY1 / AY1 &
AY2 DH / AY1 &
AY2 DX / & AY1
AY2 EH / AY1 &
AY2 EL / AY1 &
AY2 ER / AY1 &
AY2 EY / AY1 &
AY2 EYR / AY1 &
AY2 F / AY1 &
AY2 HH / AY1 &
AY2 IH / AY1 &
AY2 IR / AY1 &
AY2 IX / AY1 &
AY2 IY / AY1 &
AY2 L / AY1 &
AY2 M / AY1 &
AY2 N / AY1 &
AY2 NX / AY1 &
AY2 OR / AY1 &
AY2 OW / AY1 &
AY2 OY1 / AY1 &
AY2 R / AY1 &
AY2 S / AY1 &
AY2 SH / AY1 &
AY2 SIB / AY1 &
AY2 SIC / AY1 &
AY2 SID / AY1 &
AY2 SIG / AY1 &
AY2 SIJ / AY1 &
AY2 SIK / AY1 &
AY2 SIP / AY1 &
AY2 SIT / AY1 &
AY2 TH / AY1 &
AY2 UH / AY1 &
AY2 UR / AY1 &
AY2 UW / AY1 &
AY2 V / AY1 &
AY2 W / AY1 &
AY2 Y / AY1 &
AY2 YU / AY1 &
AY2 Z / AY1 &
AY2 ZH / AY1 &

```
B - / SIB &
B AH / SIB &
B AW / SIB &
B AY1 / SIB &
B EM / SIB &
B EY / SIB &
B IR / SIB &
B L / SIB &
B OY1 / SIB &
B UR / SIB &
B#DH / SIB &
B#L / SIB &
B#R / SIB &
B#SIB / SIB &
B#SIG / SIB &
B#SIP / SIB &
B#V / SIB &
B#Z / SIB &
CH AA / SIC &
CH AO / SIC &
CH AX / SIC &
CH EH / SIC &
CH EN / SIC &
CH EYR / SIC &
CH IX / SIC &
CH OW / SIC &
CH UR / SIC &
CH#DH / SIC &
CH#L / SIC &
CH#R / SIC &
CH#SIB / SIC &
CH#SIG / SIC &
CH#SIP / SIC &
CH#V / SIC &
CH#Z / SIC &
D AA / SID &
D AO / SID &
D AX / SID &
D EH / SID &
D EN / SID &
D EYR / SID &
D IX / SID &
D OW / SID &
D UH / SID &
D W / SID &
D#F / SID &
D#M / SID &
D#S / SID &
B AA / SIB &
B AO / SIB &
B AX / SIB &
B EH / SIB &
B EN / SIB &
B EYR / SIB &
B IX / SIB &
B OR / SIB &
B R / SIB &
B UW / SIB &
B#F / SIB &
B#M / SIB &
B#S / SIB &
B#SIC / SIB &
B#SIJ / SIB &
B#SIT / SIB &
B#W / SIB &
B#ZH / SIB &
CH AE / SIC &
CH AR / SIC &
CH AXR / SIC &
CH EL / SIC &
CH ER / SIC &
CH IH / SIC &
CH IY / SIC &
CH OY1 / SIC &
CH UW / SIC &
CH#F / SIC &
CH#M / SIC &
CH#S / SIC &
CH#SIC / SIC &
CH#SIJ / SIC &
CH#SIT / SIC &
CH#W / SIC &
CH#ZH / SIC &
D AE / SID &
D AR / SID &
D AXR / SID &
D EL / SID &
D ER / SID &
D IH / SID &
D IY / SID &
D OY1 / SID &
D UR / SID &
D YU / SID &
D#HH / SID &
D#N / SID &
D#SH / SID &
B AE / SIB &
B AR / SIB &
B AXR / SIB &
B EL / SIB &
B ER / SIB &
B IH / SIB &
B IY / SIB &
B OW / SIB &
B UH / SIB &
B YU / SIB &
B#HH / SIB &
B#N / SIB &
B#SH / SIB &
B#SID / SIB &
B#SIK / SIB &
B#TH / SIB &
B#Y / SIB &
CH - / SIC &
CH AH / SIC &
CH AW / SIC &
CH AY1 / SIC &
CH EM / SIC &
CH EY / SIC &
CH IR / SIC &
CH OR / SIC &
CH UH / SIC &
CH YU / SIC &
CH#HH / SIC &
CH#N / SIC &
CH#SH / SIC &
CH#SID / SIC &
CH#SIK / SIC &
CH#TH / SIC &
CH#Y / SIC &
D - / SID &
D AH / SID &
D AW / SID &
D AY1 / SID &
D EM / SID &
D EY / SID &
D IR / SID &
D OR / SID &
D R / SID &
D UW / SID &
D#DH / SID &
D#L / SID &
D#R / SID &
D#SIB / SID &
```

D#SIC / SID &
D#SIJ / SID &
D#SIT / SID &
D#W / SID &
D#ZH / SID &

D#SID / SID &
D#SIK / SID &
D#TH / SID &
D#Y / SID &

D#SIG / SID &
D#SIP / SID &
D#V / SID &
D#Z / SID &

DH -	DH AA	DH AE	DH AH	DH AO	DH AR
DH AW	DH AX	DH AXR	DH AY1	DH EH	DH EL
DH EM	DH EN	DH ER	DH EY	DH EYR	DH IH
DH IR	DH IX	DH IY	DH OR	DH OW	DH OY1
DH UH	DH UR	DH UW	DH YU	DH#DH	DH#F
DH#HH	DH#L	DH#M	DH#N	DH#R	DH#S
DH#SH	DH#SIB	DH#SIC	DH#SID	DH#SIG	DH#SIJ
DH#SIK	DH#SIP	DH#SIT	DH#TH	DH#V	DH#W
DH#Y	DH#Z	DH#ZH	DX AA	DX AE	DX AH
DX AO	DX AR	DX AW	DX AX	DX AXR	DX AY1
DX EH	DX EL	DX EM	DX EN	DX ER	DX EY
DX EYR	DX IH	DX IR	DX IX	DX IY	DX OR
DX OW	DX OY1	DX UH	DX UW	EH DH	EH DX
EH EL	EH F	EH HH	EH L	EH M	EH N
EH NX	EH R	EH S	EH SH	EH SIB	EH SIC
EH SID	EH SIG	EH SIJ	EH SIK	EH SIP	EH SIT
EH TH	EH V	EH W	EH Y	EH Z	EH ZH
EL -	EL AA	EL AE	EL AH	EL AO	EL AR
EL AW	EL AX	EL AXR	EL AY1	EL EH	EL EM
EL EN	EL ER	EL EY	EL EYR	EL IH	EL IR
EL IX	EL IY	EL OR	EL OW	EL OY1	EL UH
EL UR	EL UW	EL YU	EL#DH	EL#F	EL#HH
EL#L	EL#M	EL#N	EL#R	EL#S	EL#SH
EL#SIB	EL#SIC	EL#SID	EL#SIG	EL#SIJ	EL#SIK
EL#SIP	EL#SIT	EL#TH	EL#V	EL#W	EL#Y
EL#Z	EL#ZH	EM -	EM DH	EM F	EM HH
EM L	EM M	EM N	EM R	EM S	EM SH
EM SIB	EM SIC	EM SID	EM SIG	EM SIJ	EM SIK
EM SIP	EM SIT	EM TH	EM V	EM W	EM Y
EM Z	EM ZH	EN -	EN DH	EN F	EN HH
EN L	EN M	EN N	EN R	EN S	EN SH
EN SIB	EN SIC	EN SID	EN SIG	EN SIJ	EN SIK
EN SIP	EN SIT	EN TH	EN V	EN W	EN Y
EN Z	EN ZH	ER -	ER AA	ER AE	ER AH
ER AO	ER AR	ER AW	ER AX	ER AXR	ER AY1
ER DH	ER DX	ER EH	ER EL	ER ER	ER EY
ER EYR	ER F	ER HH	ER IH	ER IR	ER IX
ER IY	ER L	ER M	ER N	ER NX	ER OR
ER OW	ER OY1	ER R	ER S	ER SH	ER SIB
ER SIC	ER SID	ER SIG	ER SIJ	ER SIK	ER SIP
ER SIT	ER TH	ER UH	ER UR	ER UW	ER V
ER W	ER Y	ER YU	ER Z	ER ZH	EY -
EY AA	EY AE	EY AH	EY AO	EY AR	EY AW

```
EY AX       EY AXR      EY AY1      EY DH       EY DX       EY EH
EY EL       EY ER       EY EY       EY EYR      EY F        EY HH
EY IH       EY IR       EY IX       EY IY       EY L        EY M
EY N        EY NX       EY OR       EY OW       EY OY1      EY R
EY S        EY SH       EY SIB      EY SIC      EY SID      EY SIG
EY SIJ      EY SIK      EY SIP      EY SIT      EY TH       EY UH
EY UR       EY UW       EY V        EY W        EY Y        EY YU
EY Z        EY ZH       EYR DH      EYR DX      EYR EL      EYR F
EYR HH      EYR L       EYR M       EYR N       EYR NX      EYR R
EYR S       EYR SH      EYR SIB     EYR SIC     EYR SID     EYR SIG
EYR SIJ     EYR SIK     EYR SIP     EYR SIT     EYR TH      EYR V
EYR W       EYR Y       EYR Z       EYR ZH      F -         F AA
F AE        F AH        F AO        F AR        F AW        F AX
F AXR       F AY1       F EH        F EL        F EM        F EN
F ER        F EY        F EYR       F IH        F IR        F IX
F IY        F L         F OR        F OW        F OY1       F R
F UH        F UR        F UW        F YU        F#DH        F#F
F#HH        F#L         F#M         F#N         F#R         F#S
F#SH        F#SIB       F#SIC       F#SID       F#SIG       F#SIJ
F#SIK       F#SIP       F#SIT       F#TH        F#V         F#W
F#Y         F#Z         F#ZH
G - / SIG &             G AA / SIG &            G AE / SIG &
G AH / SIG &            G AO / SIG &            G AR / SIG &
G AW / SIG &            G AX / SIG &            G AXR / SIG &
G AY1 / SIG &           G EH / SIG &            G EL / SIG &
G EM / SIG &            G EN / SIG &            G ER / SIG &
G EY / SIG &            G EYR / SIG &           G IH / SIG &
G IR / SIG &            G IX / SIG &            G IY / SIG &
G L / SIG &             G OR / SIG &            G OW / SIG &
G OY1 / SIG &           G R / SIG &             G UH / SIG &
G UR / SIG &            G UW / SIG &            G W / SIG &
G YU / SIG &            G#DH / SIG &            G#F / SIG &
G#HH / SIG &            G#L / SIG &             G#M / SIG &
G#N / SIG &             G#R / SIG &             G#S / SIG &
G#SH / SIG &            G#SIB / SIG &           G#SIC / SIG &
G#SID / SIG &           G#SIG / SIG &           G#SIJ / SIG &
G#SIK / SIG &           G#SIP / SIG &           G#SIT / SIG &
G#TH / SIG &            G#V / SIG &             G#W / SIG &
G#Y / SIG &             G#Z / SIG &             G#ZH / SIG &
HH AA       HH AE       HH AH       HH AO       HH AR       HH AW
HH AX       HH AXR      HH AY1      HH EH       HH EL       HH EM
HH EN       HH ER       HH EY       HH EYR      HH IH       HH IR
HH IX       HH IY       HH OR       HH OW       HH OY1      HH UH
HH UR       HH UW       HH W        HH YU       IH DH       IH DX
IH EL       IH F        IH HH       IH L        IH M        IH N
IH NX       IH R        IH S        IH SH       IH SIB      IH SIC
IH SID      IH SIG      IH SIJ      IH SIK      IH SIP      IH SIT
IH TH       IH V        IH W        IH Y        IH Z        IH ZH
```

IR DH IR DX IR EL IR F IR HH IR L
IR M IR N IR NX IR R IR S IR SH
IR SIB IR SIC IR SID IR SIG IR SIJ IR SIK
IR SIP IR SIT IR TH IR V IR W IR Y
IR Z IR ZH IX DH IX EL IX F IX HH
IX L IX M IX N IX NX IX R IX S
IX SH IX SIB IX SIC IX SID IX SIG IX SIJ
IX SIK IX SIP IX SIT IX TH IX V IX W
IX Y IX Z IX ZH IY - IY AA IY AE
IY AH IY AO IY AR IY AW IY AX IY AXR
IY AY1 IY DH IY DX IY EH IY EL IY ER
IY EY IY EYR IY F IY HH IY IH IY IR
IY IX IY IY IY L IY M IY N IY NX
IY OR IY OW IY OY1 IY R IY S IY SH
IY SIB IY SIC IY SID IY SIG IY SIJ IY SIK
IY SIP IY SIT IY TH IY UH IY UR IY UW
IY V IY W IY Y IY YU IY Z IY ZH
JH - / SIJ & JH AA / SIJ & JH AE / SIJ &
JH AH / SIJ & JH AO / SIJ & JH AR / SIJ &
JH AW / SIJ & JH AX / SIJ & JH AXR / SIJ &
JH AY1 / SIJ & JH EH / SIJ & JH EL / SIJ &
JH EM / SIJ & JH EN / SIJ & JH ER / SIJ &
JH EY / SIJ & JH EYR / SIJ & JH IH / SIJ &
JH IR / SIJ & JH IX / SIJ & JH IY / SIJ &
JH OR / SIJ & JH OW / SIJ & JH OY1 / SIJ &
JH UH / SIJ & JH UR / SIJ & JH UW / SIJ &
JH YU / SIJ & JH#DH / SIJ & JH#F / SIJ &
JH#HH / SIJ & JH#L / SIJ & JH#M / SIJ &
JH#N / SIJ & JH#R / SIJ & JH#S / SIJ &
JH#SH / SIJ & JH#SIB / SIJ & JH#SIC / SIJ &
JH#SID / SIJ & JH#SIG / SIJ & JH#SIJ / SIJ &
JH#SIK / SIJ & JH#SIP / SIJ & JH#SIT / SIJ &
JH#TH / SIJ & JH#V / SIJ & JH#W / SIJ &
JH#Y / SIJ & JH#Z / SIJ & JH#ZH / SIJ &
K - / SIK & K AA / SIK & K AE / SIK &
K AH / SIK & K AO / SIK & K AR / SIK &
K AW / SIK & K AX / SIK & K AXR / SIK &
K AY1 / SIK & K EH / SIK & K EL / SIK &
K EM / SIK & K EN / SIK & K ER / SIK &
K EY / SIK & K EYR / SIK & K IH / SIK &
K IR / SIK & K IX / SIK & K IY / SIK &
K L / S SIK & K L / SIK & K OR / SIK &
K OW / SIK & K OY1 / SIK & K R / S SIK &
K R / SIK & K UH / SIK & K UR / SIK &
K UW / SIK & K W / S SIK & K W / SIK &
K YU / SIK & K#DH / SIK & K#F / SIK &
K#HH / SIK & K#L / SIK & K#M / SIK &
K#N / SIK & K#R / SIK & K#S / SIK &

K#SH / SIK &
K#SID / SIK &
K#SIK / SIK &
K#TH / SIK &
K#Y / SIK &
L AA L AE
L AX L AXR
L EN L ER
L IX L IY
L UR L UW
M AH M AO
M AY1 M EH
M EY M EYR
M OR M OW
M YU M#DH
M#N M#R
M#SID M#SIG
M#TH M#V
N - N AA
N AW N AX
N EL N EM
N IH N IR
N OY1 N UH
N#F N#HH
N#S N#SH
N#SIJ N#SIK
N#W N#Y
NX AE NX AH
NX AXR NX AY1
NX ER NX EY
NX IY NX OR
NX UW NX YU
NX#M NX#N
NX#SIC NX#SID
NX#SIT NX#TH
NX#ZH OR DH
OR L OR M
OR SH OR SIB
OR SIK OR SIP
OR Y OR Z
OW AH OW AO
OW AY1 OW DH
OW EY OW EYR
OW IX OW IY
OW OR OW OW
OW SIB OW SIC
OW SIP OW SIT
OW V OW W

K#SIB / SIK &
K#SIG / SIK &
K#SIP / SIK &
K#V / SIK &
K#Z / SIK &
L AH L AO
L AY1 L EH
L EY L EYR
L OR L OW
L YU M -
M AR M AW
M EL M EM
M IH M IR
M OY1 M UH
M#F M#HH
M#S M#SH
M#SIJ M#SIK
M#W M#Y
N AE N AH
N AXR N AY1
N EN N ER
N IX N IY
N UR N UW
N#L N#M
N#SIB N#SIC
N#SIP N#SIT
N#Z N#ZH
NX AO NX AR
NX EH NX EL
NX EYR NX IH
NX OW NX OY1
NX#DH NX#F
NX#R NX#S
NX#SIG NX#SIJ
NX#V NX#W
OR DX OR EL
OR N OR NX
OR SIC OR SID
OR SIT OR TH
OR ZH OW -
OW AR OW AW
OW DX OW EH
OW F OW HH
OW L OW M
OW OY1 OW R
OW SID OW SIG
OW TH OW UH
OW Y OW YU

K#SIC / SIK &
K#SIJ / SIK &
K#SIT / SIK &
K#W / SIK &
K#ZH / SIK &
L AR L AW
L EL L EM
L IH L IR
L OY1 L UH
M AA M AE
M AX M AXR
M EN M ER
M IX M IY
M UR M UW
M#L M#M
M#SIB M#SIC
M#SIP M#SIT
M#Z M#ZH
N AO N AR
N DX N EH
N EY N EYR
N OR N OW
N YU N#DH
N#N N#R
N#SID N#SIG
N#TH N#V
NX - NX AA
NX AW NX AX
NX EM NX EN
NX IR NX IX
NX UH NX UR
NX#HH NX#L
NX#SH NX#SIB
NX#SIK NX#SIP
NX#Y NX#Z
OR F OR HH
OR R OR S
OR SIG OR SIJ
OR V OR W
OW AA OW AE
OW AX OW AXR
OW EL OW ER
OW IH OW IR
OW N OW NX
OW S OW SH
OW SIJ OW SIK
OW UR OW UW
OW Z OW ZH

OY1 OY2 / & -
OY1 OY2 / & AH
OY1 OY2 / & AW
OY1 OY2 / & AY1
OY1 OY2 / & EH
OY1 OY2 / & EY
OY1 OY2 / & HH
OY1 OY2 / & IX
OY1 OY2 / & M
OY1 OY2 / & OR
OY1 OY2 / & R
OY1 OY2 / & SIB
OY1 OY2 / & SIG
OY1 OY2 / & SIP
OY1 OY2 / & UH
OY1 OY2 / & V
OY1 OY2 / & YU
OY2 - / OY1 &
OY2 AH / OY1 &
OY2 AW / OY1 &
OY2 AY1 / OY1 &
OY2 EH / OY1 &
OY2 EY / OY1 &
OY2 HH / OY1 &
OY2 IX / OY1 &
OY2 M / OY1 &
OY2 OR / OY1 &
OY2 R / OY1 &
OY2 SIB / OY1 &
OY2 SIG / OY1 &
OY2 SIP / OY1 &
OY2 UH / OY1 &
OY2 V / OY1 &
OY2 YU / OY1 &
P - / SIP &
P AH / SIP &
P AW / SIP &
P AY1 / SIP &
P EM / SIP &
P EY / SIP &
P IR / SIP &
P L / S SIP &
P OW / SIP &
P R / SIP &
P UW / SIP &
P#F / SIP &
P#M / SIP &
P#S / SIP &

OY1 OY2 / & AA
OY1 OY2 / & AO
OY1 OY2 / & AX
OY1 OY2 / & DH
OY1 OY2 / & EL
OY1 OY2 / & EYR
OY1 OY2 / & IH
OY1 OY2 / & IY
OY1 OY2 / & N
OY1 OY2 / & OW
OY1 OY2 / & S
OY1 OY2 / & SIC
OY1 OY2 / & SIJ
OY1 OY2 / & SIT
OY1 OY2 / & UR
OY1 OY2 / & W
OY1 OY2 / & Z
OY2 AA / OY1 &
OY2 AO / OY1 &
OY2 AX / OY1 &
OY2 DH / OY1 &
OY2 EL / OY1 &
OY2 EYR / OY1 &
OY2 IH / OY1 &
OY2 IY / OY1 &
OY2 N / OY1 &
OY2 OW / OY1 &
OY2 S / OY1 &
OY2 SIC / OY1 &
OY2 SIJ / OY1 &
OY2 SIT / OY1 &
OY2 UR / OY1 &
OY2 W / OY1 &
OY2 Z / OY1 &
P AA / SIP &
P AO / SIP &
P AX / SIP &
P EH / SIP &
P EN / SIP &
P EYR / SIP &
P IX / SIP &
P L / SIP &
P OY1 / SIP &
P UH / SIP &
P YU / SIP &
P#HH / SIP &
P#N / SIP &
P#SH / SIP &

OY1 OY2 / & AE
OY1 OY2 / & AR
OY1 OY2 / & AXR
OY1 OY2 / & DX
OY1 OY2 / & ER
OY1 OY2 / & F
OY1 OY2 / & IR
OY1 OY2 / & L
OY1 OY2 / & NX
OY1 OY2 / & OY1
OY1 OY2 / & SH
OY1 OY2 / & SID
OY1 OY2 / & SIK
OY1 OY2 / & TH
OY1 OY2 / & UW
OY1 OY2 / & Y
OY1 OY2 / & ZH
OY2 AE / OY1 &
OY2 AR / OY1 &
OY2 AXR / OY1 &
OY2 DX / OY1 &
OY2 ER / OY1 &
OY2 F / OY1 &
OY2 IR / OY1 &
OY2 L / OY1 &
OY2 NX / OY1 &
OY2 OY1 / OY1 &
OY2 SH / OY1 &
OY2 SID / OY1 &
OY2 SIK / OY1 &
OY2 TH / OY1 &
OY2 UW / OY1 &
OY2 Y / OY1 &
OY2 ZH / OY1 &
P AE / SIP &
P AR / SIP &
P AXR / SIP &
P EL / SIP &
P ER / SIP &
P IH / SIP &
P IY / SIP &
P OR / SIP &
P R / S SIP &
P UR / SIP &
P#DH / SIP &
P#L / SIP &
P#R / SIP &
P#SIB / SIP &

```
P#SIC / SIP &          P#SID / SIP &          P#SIG / SIP &
P#SIJ / SIP &          P#SIK / SIP &          P#SIP / SIP &
P#SIT / SIP &          P#TH / SIP &           P#V / SIP &
P#W / SIP &            P#Y / SIP &            P#Z / SIP &
P#ZH / SIP &
R -          R AA      R AE       R AH        R AO       R AR
R AW         R AX      R AXR      R AY1       R EH       R EL
R EM         R EN      R ER       R EY        R EYR      R IH
R IR         R IX      R IY       R OR        R OW       R OY1
R UH         R UR      R UW       R#DH        R#F        R#HH
R#L          R#M       R#N        R#R         R#S        R#SH
R#SIB        R#SIC     R#SID      R#SIG       R#SIJ      R#SIK
R#SIP        R#SIT     R#TH       R#V         R#W        R#Y
R#Z          R#ZH      S -        S AA        S AE       S AH
S AO         S AR      S AW       S AX        S AXR      S AY1
S EH         S EL      S EM       S EN        S ER       S EY
S EYR        S IH      S IR       S IX        S IY       S L
S M          S N       S OR       S OW        S OY1      S SIK
S SIK / & K L          S SIK / & K R          S SIK / & K W
S SIP
S SIP / & P L          S SIP / & P R
S SIT
S SIT / & T R
S UH         S UR      S UW       S W         S YU       S#DH
S#F          S#HH      S#L        S#M         S#N        S#R
S#S          S#SH      S#SIB      S#SIC       S#SID      S#SIG
S#SIJ        S#SIK     S#SIP      S#SIT       S#TH       S#V
S#W          S#Y       S#Z        S#ZH        SH -       SH AA
SH AE        SH AH     SH AO      SH AR       SH AW      SH AX
SH AXR       SH AY1    SH EH      SH EL       SH EM      SH EN
SH ER        SH EY     SH EYR     SH IH       SH IR      SH IX
SH IY        SH OR     SH OW      SH OY1      SH R       SH UH
SH UR        SH UW     SH YU      SH#DH       SH#F       SH#HH
SH#L         SH#M      SH#N       SH#R        SH#S       SH#SH
SH#SIB       SH#SIC    SH#SID     SH#SIG      SH#SIJ     SH#SIK
SH#SIP       SH#SIT    SH#TH      SH#V        SH#W       SH#Y
SH#Z         SH#ZH
SIB B / & # DH         SIB B / & # F          SIB B / & # HH
SIB B / & # L          SIB B / & # M          SIB B / & # N
SIB B / & # R          SIB B / & # S          SIB B / & # SH
SIB B / & # SIB        SIB B / & # SIC        SIB B / & # SID
SIB B / & # SIG        SIB B / & # SIJ        SIB B / & # SIK
SIB B / & # SIP        SIB B / & # SIT        SIB B / & # TH
SIB B / & # V          SIB B / & # W          SIB B / & # Y
SIB B / & # Z          SIB B / & # ZH         SIB B / & -
SIB B / & AA           SIB B / & AE           SIB B / & AH
SIB B / & AO           SIB B / & AR           SIB B / & AW
SIB B / & AX           SIB B / & AXR          SIB B / & AY1
```

```
SIB B / & EH
SIB B / & EN
SIB B / & EYR
SIB B / & IX
SIB B / & OR
SIB B / & R
SIB B / & UW
SIC CH / & # F
SIC CH / & # M
SIC CH / & # S
SIC CH / & # SIC
SIC CH / & # SIJ
SIC CH / & # SIT
SIC CH / & # W
SIC CH / & # ZH
SIC CH / & AE
SIC CH / & AR
SIC CH / & AXR
SIC CH / & EL
SIC CH / & ER
SIC CH / & IH
SIC CH / & IY
SIC CH / & OY1
SIC CH / & UW
SID D / & # F
SID D / & # M
SID D / & # S
SID D / & # SIC
SID D / & # SIJ
SID D / & # SIT
SID D / & # W
SID D / & # ZH
SID D / & AE
SID D / & AR
SID D / & AXR
SID D / & EL
SID D / & ER
SID D / & IH
SID D / & IY
SID D / & OY1
SID D / & UR
SID D / & YU
SIG G / & # HH
SIG G / & # N
SIG G / & # SH
SIG G / & # SID
SIG G / & # SIK
SIG G / & # TH
SIB B / & EL
SIB B / & ER
SIB B / & IH
SIB B / & IY
SIB B / & OW
SIB B / & UH
SIB B / & YU
SIC CH / & # HH
SIC CH / & # N
SIC CH / & # SH
SIC CH / & # SID
SIC CH / & # SIK
SIC CH / & # TH
SIC CH / & # Y
SIC CH / & -
SIC CH / & AH
SIC CH / & AW
SIC CH / & AY1
SIC CH / & EM
SIC CH / & EY
SIC CH / & IR
SIC CH / & OR
SIC CH / & UH
SIC CH / & YU
SID D / & # HH
SID D / & # N
SID D / & # SH
SID D / & # SID
SID D / & # SIK
SID D / & # TH
SID D / & # Y
SID D / & -
SID D / & AH
SID D / & AW
SID D / & AY1
SID D / & EM
SID D / & EY
SID D / & IR
SID D / & OR
SID D / & R
SID D / & UW
SIG G / & # DH
SIG G / & # L
SIG G / & # R
SIG G / & # SIB
SIG G / & # SIG
SIG G / & # SIP
SIG G / & # V
SIB B / & EM
SIB B / & EY
SIB B / & IR
SIB B / & L
SIB B / & OY1
SIB B / & UR
SIC CH / & # DH
SIC CH / & # L
SIC CH / & # R
SIC CH / & # SIB
SIC CH / & # SIG
SIC CH / & # SIP
SIC CH / & # V
SIC CH / & # Z
SIC CH / & AA
SIC CH / & AO
SIC CH / & AX
SIC CH / & EH
SIC CH / & EN
SIC CH / & EYR
SIC CH / & IX
SIC CH / & OW
SIC CH / & UR
SID D / & # DH
SID D / & # L
SID D / & # R
SID D / & # SIB
SID D / & # SIG
SID D / & # SIP
SID D / & # V
SID D / & # Z
SID D / & AA
SID D / & AO
SID D / & AX
SID D / & EH
SID D / & EN
SID D / & EYR
SID D / & IX
SID D / & OW
SID D / & UH
SID D / & W
SIG G / & # F
SIG G / & # M
SIG G / & # S
SIG G / & # SIC
SIG G / & # SIJ
SIG G / & # SIT
SIG G / & # W
```

```
SIG G / & # Y
SIG G / & -
SIG G / & AH
SIG G / & AW
SIG G / & AYl
SIG G / & EM
SIG G / & EY
SIG G / & IR
SIG G / & L
SIG G / & OYl
SIG G / & UR
SIG G / & YU
SIJ JH / & # HH
SIJ JH / & # N
SIJ JH / & # SH
SIJ JH / & # SID
SIJ JH / & # SIK
SIJ JH / & # TH
SIJ JH / & # Y
SIJ JH / & -
SIJ JH / & AH
SIJ JH / & AW
SIJ JH / & AYl
SIJ JH / & EM
SIJ JH / & EY
SIJ JH / & IR
SIJ JH / & OR
SIJ JH / & UH
SIJ JH / & YU
SIK K / & # HH
SIK K / & # N
SIK K / & # SH
SIK K / & # SID
SIK K / & # SIK
SIK K / & # TH
SIK K / & # Y
SIK K / & -
SIK K / & AH
SIK K / & AW
SIK K / & AYl
SIK K / & EM
SIK K / & EY
SIK K / & IR
SIK K / & L
SIK K / & OYl
SIK K / & UR
SIK K / & YU
SIK K / S & W
SIG G / & # Z
SIG G / & AA
SIG G / & AO
SIG G / & AX
SIG G / & EH
SIG G / & EN
SIG G / & EYR
SIG G / & IX
SIG G / & OR
SIG G / & R
SIG G / & UW
SIJ JH / & # DH
SIJ JH / & # L
SIJ JH / & # R
SIJ JH / & # SIB
SIJ JH / & # SIG
SIJ JH / & # SIP
SIJ JH / & # V
SIJ JH / & # Z
SIJ JH / & AA
SIJ JH / & AO
SIJ JH / & AX
SIJ JH / & EH
SIJ JH / & EN
SIJ JH / & EYR
SIJ JH / & IX
SIJ JH / & OW
SIJ JH / & UR
SIK K / & # DH
SIK K / & # L
SIK K / & # R
SIK K / & # SIB
SIK K / & # SIG
SIK K / & # SIP
SIK K / & # V
SIK K / & # Z
SIK K / & AA
SIK K / & AO
SIK K / & AX
SIK K / & EH
SIK K / & EN
SIK K / & EYR
SIK K / & IX
SIK K / & OR
SIK K / & R
SIK K / & UW
SIK K / S & L
SIP P / & # DH
SIG G / & # ZH
SIG G / & AE
SIG G / & AR
SIG G / & AXR
SIG G / & EL
SIG G / & ER
SIG G / & IH
SIG G / & IY
SIG G / & OW
SIG G / & UH
SIG G / & W
SIJ JH / & # F
SIJ JH / & # M
SIJ JH / & # S
SIJ JH / & # SIC
SIJ JH / & # SIJ
SIJ JH / & # SIT
SIJ JH / & # W
SIJ JH / & # ZH
SIJ JH / & AE
SIJ JH / & AR
SIJ JH / & AXR
SIJ JH / & EL
SIJ JH / & ER
SIJ JH / & IH
SIJ JH / & IY
SIJ JH / & OYl
SIJ JH / & UW
SIK K / & # F
SIK K / & # M
SIK K / & # S
SIK K / & # SIC
SIK K / & # SIJ
SIK K / & # SIT
SIK K / & # W
SIK K / & # ZH
SIK K / & AE
SIK K / & AR
SIK K / & AXR
SIK K / & EL
SIK K / & ER
SIK K / & IH
SIK K / & IY
SIK K / & OW
SIK K / & UH
SIK K / & W
SIK K / S & R
SIP P / & # F
```

SIP P / & # HH
SIP P / & # N
SIP P / & # SH
SIP P / & # SID
SIP P / & # SIK
SIP P / & # TH
SIP P / & # Y
SIP P / & -
SIP P / & AH
SIP P / & AW
SIP P / & AY1
SIP P / & EM
SIP P / & EY
SIP P / & IR
SIP P / & L
SIP P / & OY1
SIP P / & UR
SIP P / S & L
SIT T / & # F
SIT T / & # M
SIT T / & # S
SIT T / & # SIC
SIT T / & # SIJ
SIT T / & # SIT
SIT T / & # W
SIT T / & # ZH
SIT T / & AE
SIT T / & AR
SIT T / & AXR
SIT T / & EL
SIT T / & ER
SIT T / & IH
SIT T / & IY
SIT T / & OY1
SIT T / & UR
SIT T / & YU
T AA / SIT &
T AO / SIT &
T AX / SIT &
T EH / SIT &
T EN / SIT &
T EYR / SIT &
T IX / SIT &
T OW / SIT &
T R / SIT &
T UW / SIT &
T#DH / SIT &
T#L / SIT &

SIP P / & # L
SIP P / & # R
SIP P / & # SIB
SIP P / & # SIG
SIP P / & # SIP
SIP P / & # V
SIP P / & # Z
SIP P / & AA
SIP P / & AO
SIP P / & AX
SIP P / & EH
SIP P / & EN
SIP P / & EYR
SIP P / & IX
SIP P / & OR
SIP P / & R
SIP P / & UW
SIP P / S & R
SIT T / & # HH
SIT T / & # N
SIT T / & # SH
SIT T / & # SID
SIT T / & # SIK
SIT T / & # TH
SIT T / & # Y
SIT T / & -
SIT T / & AH
SIT T / & AW
SIT T / & AY1
SIT T / & EM
SIT T / & EY
SIT T / & IR
SIT T / & OR
SIT T / & R
SIT T / & UW
SIT T / S & R
T AE / SIT &
T AR / SIT &
T AXR / SIT &
T EL / SIT &
T ER / SIT &
T IH / SIT &
T IY / SIT &
T OY1 / SIT &
T UH / SIT &
T W / SIT &
T#F / SIT &
T#M / SIT &

SIP P / & # M
SIP P / & # S
SIP P / & # SIC
SIP P / & # SIJ
SIP P / & # SIT
SIP P / & # W
SIP P / & # ZH
SIP P / & AE
SIP P / & AR
SIP P / & AXR
SIP P / & EL
SIP P / & ER
SIP P / & IH
SIP P / & IY
SIP P / & OW
SIP P / & UH
SIP P / & YU
SIT T / & # DH
SIT T / & # L
SIT T / & # R
SIT T / & # SIB
SIT T / & # SIG
SIT T / & # SIP
SIT T / & # V
SIT T / & # Z
SIT T / & AA
SIT T / & AO
SIT T / & AX
SIT T / & EH
SIT T / & EN
SIT T / & EYR
SIT T / & IX
SIT T / & OW
SIT T / & UH
SIT T / & W
T - / SIT &
T AH / SIT &
T AW / SIT &
T AY1 / SIT &
T EM / SIT &
T EY / SIT &
T IR / SIT &
T OR / SIT &
T R / S SIT &
T UR / SIT &
T YU / SIT &
T#HH / SIT &
T#N / SIT &

T#R / SIT &
T#SIB / SIT &
T#SIG / SIT &
T#SIP / SIT &
T#V / SIT &
T#Z / SIT &

TH -
TH AW
TH EM
TH IR
TH R
TH#DH
TH#R
TH#SIG
TH#V
TQ EM
UH HH
UH S
UH SIJ
UH W
UR EL
UR NX
UR SID
UR TH
UW -
UW AW
UW EH
UW HH
UW M
UW R
UW SIG
UW UH
UW YU
V AH
V AY1
V EY
V OR
V YU
V#N
V#SID
V#TH
W AA
W AX
W EN
W IX
W UR
Y AR
Y EL

TH AA
TH AX
TH EN
TH IX
TH UH
TH#F
TH#S
TH#SIJ
TH#W
TQ EN
UH L
UH SH
UH SIK
UH Y
UR F
UR R
UR SIG
UR V
UW AA
UW AX
UW EL
UW IH
UW N
UW S
UW SIJ
UW UR
UW Z
V AO
V EH
V EYR
V OW
V#DH
V#R
V#SIG
V#V
W AE
W AXR
W ER
W IY
W UW
Y AW
Y EM

T#S / SIT &
T#SIC / SIT &
T#SIJ / SIT &
T#SIT / SIT &
T#W / SIT &
T#ZH / SIT &

TH AE
TH AXR
TH ER
TH IY
TH UR
TH#HH
TH#SH
TH#SIK
TH#Y
UH DH
UH M
UH SIB
UH SIP
UH Z
UR HH
UR S
UR SIJ
UW W
UW AE
UW AXR
UW ER
UW IR
UW NX
UW SH
UW SIK
UW UW
UW ZH
V AR
V EL
V IH
V OY1
V#F
V#S
V#SIJ
V#W
W AH
W AY1
W EY
W OR
Y AA
Y AX
Y EN

TH AH
TH AY1
TH EY
TH OR
TH UW
TH#L
TH#SIB
TH#SIP
TH#Z
UH DX
UH N
UH SIC
UH SIT
UH ZH
UR L
UR SH
UR SIK
UR Y
UW AH
UW AY1
UW EY
UW IX
UW OR
UW SIB
UW SIP
UW V
V -
V AW
V EM
V IR
V UH
V#HH
V#SH
V#SIK
V#Y
W AO
W EH
W EYR
W OW
Y AE
Y AXR
Y ER

T#SH / SIT &
T#SID / SIT &
T#SIK / SIT &
T#TH / SIT &
T#Y / SIT &

TH AO
TH EH
TH EYR
TH OW
TH W
TH#M
TH#SIC
TH#SIT
TH#ZH
UH EL
UH NX
UH SID
UH TH
UR DH
UR M
UR SIB
UR SIP
UR Z
UW AO
UW DH
UW EYR
UW IY
UW OW
UW SIC
UW SIT
UW W
V AA
V AX
V EN
V IX
V UR
V#L
V#SIB
V#SIP
V#Z
W AR
W EL
W IH
W OY1
Y AH
Y AY1
Y EY

TH AR
TH EL
TH IH
TH OY1
TH YU
TH#N
TH#SID
TH#TH
TQ EL
UH F
UH R
UH SIG
UH V
UR DX
UR N
UR SIC
UR SIT
UR ZH
UW AR
UW DX
UW F
UW L
UW OY1
UW SID
UW TH
UW Y
V AE
V AXR
V ER
V IY
V UW
V#M
V#SIC
V#SIT
V#ZH
W AW
W EM
W IR
W UH
Y AO
Y EH
Y EYR

Y IH	Y IR	Y IX	Y IY	Y OR	Y OW
Y OYl	Y UH	Y UR	Y UW	YU DH	YU EL
YU F	YU HH	YU L	YU M	YU N	YU NX
YU S	YU SH	YU SIB	YU SIC	YU SID	YU SIG
YU SIJ	YU SIK	YU SIP	YU SIT	YU TH	YU V
YU Z	YU ZH	Z -	Z AA	Z AE	Z AH
Z AO	Z AR	Z AW	Z AX	Z AXR	Z AYl
Z EH	Z EL	Z EM	Z EN	Z ER	Z EY
Z EYR	Z IH	Z IR	Z IX	Z IY	Z OR
Z OW	Z OYl	Z UH	Z UR	Z UW	Z YU
Z#DH	Z#F	Z#HH	Z#L	Z#M	Z#N
Z#R	Z#S	Z#SH	Z#SIB	Z#SIC	Z#SID
Z#SIG	Z#SIJ	Z#SIK	Z#SIP	Z#SIT	Z#TH
Z#V	Z#W	Z#Y	Z#Z	Z#ZH	ZH -
ZH AA	ZH AE	ZH AH	ZH AO	ZH AR	ZH AW
ZH AX	ZH AXR	ZH AYl	ZH EH	ZH EL	ZH EM
ZH EN	ZH ER	ZH EY	ZH EYR	ZH IH	ZH IR
ZH IX	ZH IY	ZH OR	ZH OW	ZH OYl	ZH UH
ZH UR	ZH UW	ZH YU	ZH#DH	ZH#F	ZH#HH
ZH#L	ZH#M	ZH#N	ZH#R	ZH#S	ZH#SH
ZH#SIB	ZH#SIC	ZH#SID	ZH#SIG	ZH#SIJ	ZH#SIK
ZH#SIP	ZH#SIT	ZH#TH	ZH#V	ZH#W	ZH#Y
ZH#Z	ZH#ZH				

APPENDIX C

AN ADAPTIVE-TRANSFORM BASEBAND CODER

(Proceedings of the 97th Meeting of the Acoustical Society of America, pp. 377-380, June 1979)

AD-A092 578 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
SPEECH COMPRESSION AND SYNTHESIS.(U)
OCT 80 M BEROUTI, J MAKOUL, R SCHWARTZ F19628-78-C-0136
UNCLASSIFIED BBN-4414 NL

2 OF 2
AD A092578

END
DATE FILMED
2-81
DTIC

AN ADAPTIVE-TRANSFORM BASEBAND CODER

Michael Berouti and John Makhoul Bolt Beranek and Newman Inc., Cambridge, MA 02138

In a baseband coder, the baseband may be coded in a number of different ways. In this paper, we describe coding of the baseband LPC residual using adaptive transform coding (ATC). At the receiver, high-frequency regeneration is accomplished by our recently developed method of spectral duplication. Care is taken to ensure that the harmonic structure is not interrupted as a result of the spectral duplication process. The results are compared to time-domain coding of the baseband residual.

Introduction

Baseband coders, or what are known also as voice-excited coders [1-6], were originally proposed as a compromise between pitch-excited coders (such as LPC, channel and homomorphic vocoders) and waveform coders. Today, baseband coders offer attractive alternatives at data rates in the range 6.4-9.6 kb/s. This range of data rates has become increasingly important because modems are now available that operate reliably in that range over regular telephone lines.

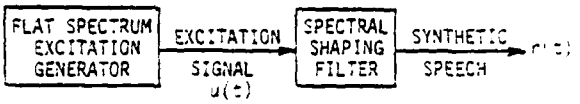


Fig. 1 Basic synthesis model for baseband coder.

Below, we shall assume that, at the receiver, the synthesizer obeys the general synthesis model shown in Fig. 1. In the figure, the synthetic or reconstructed speech signal r(t) is generated as the result of applying a time-varying excitation signal u(t) as input to a time-varying spectral shaping filter. The spectral envelope of the excitation is assumed to be flat, so that the spectral envelope of the synthetic speech is determined completely by the spectral shaping filter. The parameters of the model, i.e., the excitation and the filter, must be computed and transmitted periodically by the transmitter. Those parameters that represent the speech spectrum, denoted as spectral parameters, are computed, quantized and transmitted every 15-30 ms. In a baseband coder, a low-frequency portion of the excitation, known as a baseband, is transmitted and used at the receiver to regenerate the high-frequency portion of the excitation. The sum of the transmitted baseband and the regenerated high-frequency band constitute the excitation u(t) to the synthesizer.

In a baseband coder, synthetic speech quality is determined by four factors: a) width of the baseband, b) coding of the baseband, c) estimation and coding of spectral parameters, and d) the high-frequency regeneration (HFR) method employed. In this paper we shall concentrate mainly on the second and fourth factors.

A Residual-Excited Baseband Coder

Figure 2 shows the transmitter portion of a digital baseband coder, based on a linear prediction (LPC) representation. The speech signal s(t), sampled at 2W Hz, is filtered with the LPC inverse filter A(z) to produce the residual waveform e(t). The subsequent coding of e(t) may be quite simple, using adaptive quantization (APCM), or may be more complicated, employing adaptive predictive coding (APC), or sub-band coding (SBC) techniques. In this paper, we propose the use of Adaptive Transform Coding (ATC) techniques [9,10,11] to code the baseband residual.

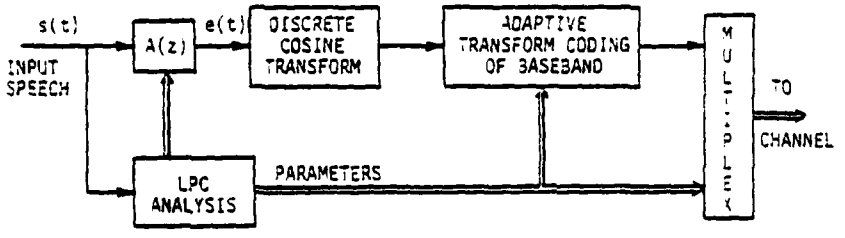


Fig. 2 Transmitter for adaptive-transform baseband coder.

As shown in Fig. 2, the discrete cosine transform (DCT) of the residual e(t) is obtained and coded using ATC techniques. The necessary lowpass (or bandpass) operation needed to retain the low frequency portion of the residual is not shown explicitly in the figure, since it can be done directly in the cosine transform domain. Only the baseband portion of the transform is coded and

transmitted. The method of coding is explained in the next section. At the receiver (see Fig. 3), the tasks are to regenerate the high-frequency portion of the DCT of the excitation signal, perform an inverse cosine transformation, and synthesize the output speech, using the model of Fig. 1.

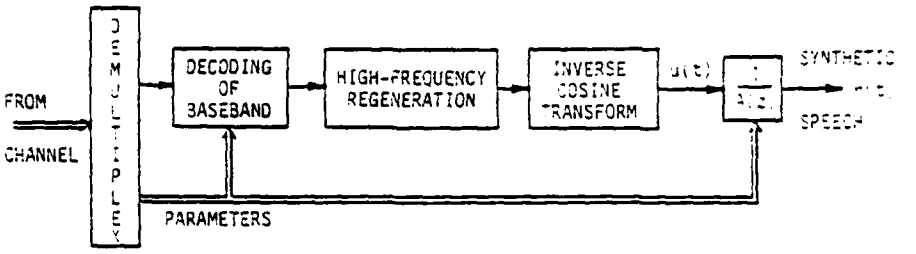


Fig. 3 Receiver for adaptive-transform baseband coder.

Transform Coding of the Baseband

Adaptive transform coding has been recently shown to be an efficient method for digital transmission of the speech signal [9, 10,11]. However, the available ATC methods deal with the transform of the speech signal itself rather than that of the residual. There are several reasons to prefer using ATC to code the residual rather than the speech signal. One important reason is that the synthesis filter 1/A(z) (see Fig. 3) smooths the possible frame-boundary discontinuities caused by quantization. Thus no overlap between frames is necessary. We have verified the above statement experimentally for the case of the baseband coder. In the future, we hope to reach a similar conclusion for the full-band case. Other advantages in using ATC to code the baseband residual are mentioned later in this section.

ATC of speech. Here, we briefly review ATC of speech and give it a new interpretation, which we use to introduce the changes that are necessary for using ATC to code the residual.

In quantizing schemes, an often used performance criterion is to maximize the signal-to-quantization-noise (S/Q) ratio, for a fixed number of bits. It can be shown that the S/Q ratio is maximized by quantizing each of the transform (DCT) coefficients using the same quantization step size, and where the available bits are properly allocated among the coefficients. The allocation of bits (or quantizer levels) depends on the model of the DCT spectrum that is employed. It can be shown that, when the number of levels used is set to be proportional to the magnitude of the spectral model, the gain in S/Q afforded by ATC over APCM of speech, is equal to the ratio of the arithmetic mean to the geometric mean of the model. For speech, a good model to use consists of a smooth spectral (LPC) envelope and a harmonic structure to model pitch [10].

An alternate way of viewing the quantization process in ATC is to assume that we first "normalize" the DCT coefficients by dividing them by the corresponding magnitude of the spectral model. Then, to achieve the same S/Q as before, we must use the same levels (number of bits) as in the non-normalized case, but we must vary the quantization step-size for each coefficient to be inversely proportional to the spectral model amplitudes at that frequency. This alternate view of ATC paves the way for using ATC in coding the residual.

ATC of the residual. The DCT of the residual can be considered to be equivalent to the "normalized" DCT of speech. The reason is that inverse filtering in the time domain is approximately equivalent in the transform domain to dividing the DCT coefficients of the speech by the magnitude of the LPC spectral envelope. Therefore, by using the same number of bits allocated at each frequency as in the usual ATC case, and varying the step-size for each DCT coefficient in proportion to the magnitude spectrum of the LPC inverse filter, one maintains the same S/Q ratio as in ATC of the speech.

In recent experiments [12] with the baseband coder, we used APCM to code the baseband residual. In these experiments, we were able to determine that some of the roughness in the reconstructed speech signal was due to quantization noise. We feel that in our proposed baseband coder, ATC of the baseband will provide a sufficient increase in S/Q over APCM to help eliminate the roughness caused by quantization noise. We shall also incorporate spectral noise shaping into the design of the system, as has been done on the full-band speech [10,11].

We note here that the proposed scheme is a frequency-domain coder, and thus, it lends itself nicely to the pitch-adaptive HFR method to be discussed in the next section. A further advantage of the proposed scheme is that it is applicable to the full-band residual, or any portion thereof. Thus, it lends itself quite easily to situations where a multirate system is desired. For the full-band case, the transmission rate is 16 kb/s. For lower rates, only a baseband is

transmitted. The width of the baseband is determined by the channel bandwidth. Various bit-rates are achieved by increasing or decreasing the number of transmitted DCT coefficients, i.e., wider or narrower basebands. We plan to test our scheme in a multirate environment.

High Frequency Regeneration

It is well-known that if the baseband has either the voice fundamental or at least two adjacent harmonics, a waveform containing all the harmonics of voiced input speech can be generated by feeding the baseband signal to an instantaneous, zero-memory, nonlinear device. A traditional approach used in the past to provide the requisite nonlinearity has been some form of waveform rectification. We have presented elsewhere [13] a new HFR method based on duplication of the baseband spectrum. In particular, we presented time-domain systems that perform spectral duplication in each of two ways: (a) Spectral folding, and (b) spectral translation. Fig. 4 illustrates the effects of the two methods. In the figure, W denotes the total input bandwidth and B denotes the width of the baseband, with W=LB, where L is an integer.

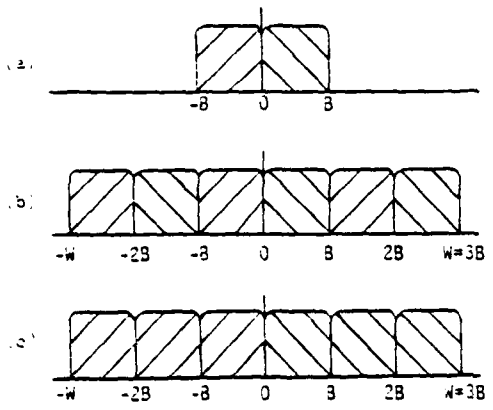


Fig. 4 a) Baseband spectrum
b) 3-band spectral folding
c) 3-band spectral translation.

Our experience with the above mentioned integer-band spectral duplication methods is that they do not seem to cause any perceptible roughness, as is the case with waveform rectification methods. However, some low-level background tones are audible with the new HFR methods. A possible reason for the existence of these background tones is the fact that the harmonic structure is interrupted at multiples of the baseband width B Hz. Therefore, we hypothesized that the tones could be eliminated by adjusting the width B of the baseband to be a multiple of the pitch fundamental frequency on a short-term basis. Such a scheme would require an enormous amount of computation if it were to be implemented in the time domain. Below, we shall explain a frequency-domain pitch-adaptive spectral translation method.

The idea here is based on the fact that, in the adaptive-transform baseband coder, the baseband DCT components can be easily duplicated at higher frequencies, to obtain the full-band excitation signal. In order not to interrupt the harmonic structure of the excitation signal, an estimate of the pitch must be used. In case the spectral model at the transmitter makes use of pitch, the value of pitch is transmitted and is readily available at the receiver. Otherwise, the receiver can easily extract a pitch value from the received baseband, e.g., by detecting the location of the peak of the autocorrelation of the baseband. With pitch known, the receiver extracts a subinterval of the baseband containing an integer number of harmonics. The chosen subinterval is duplicated (translated) at higher frequencies, as many times as necessary, to fill the missing frequency components.

For voiced sounds, we found that good perceptual results are obtained when the subinterval extends from the spectral valley just before the first harmonic to the valley just after the last complete harmonic present in the baseband. The upper frequency edge of the subinterval is C Hz, and is pitch-dependent. For unvoiced sounds, the subinterval consists of the whole baseband, less its two end points: the d.c. component and the component at B Hz. Following the HFR process, an inverse DCT yields the full-band time-domain excitation signal to be applied to the synthesis filter 1/A(z). We note here that the effective baseband width is C<B. The received frequency components between C and B are discarded, and those between C and W are regenerated.

One possible extension of the above described HFR method is to let the transmitter locate and extract the subinterval of the baseband. In such a case the transmitted baseband would be pitch-adaptive and of width C Hz.

A second, and perceptually more important extension of the method, provides for a better placement of the frequency-translated intervals. In the above described method of HFR, we assumed that the spectrum of voiced sounds is periodic in frequency. However, we know that speech spectra are not exactly harmonic in structure. To take into account the irregularities of the speech spectrum, we shift the high-frequency interval around its nominal position, in such a manner as to match, as best as possible, the corresponding original DCT components of the full-band residual. This task is done at the transmitter where the DCT of the full-band is available. First, the chosen subinterval is translated to its nominal high-frequency position. Then, it is cross-correlated with the corresponding DCT components of the full-band residual. The "optimal" location is then chosen to be at the positive maximum value of the cross-correlation. When short correlation lags are considered, i.e., between 0 and 3, the additional cost is only 2 bits for

each frequency-translated interval. The transmitted information indicates to the receiver where to place the baseband subinterval in the high frequency region, relative to its nominal position.

Preliminary Results

We have implemented a preliminary version of the proposed adaptive-transform baseband coder and simulated its operation at a transmission rate of 9.6 kb/s. For inputs at a sampling rate of 6.67 kHz, we have chosen the frame size to be 19.2 ms, i.e., 128 samples. At each frame, a 128-point DCT of the residual is obtained, and only the first 48 coefficients (3.1 kHz) are retained. In our experiments thus far, we transmitted the baseband residual using APCM. Parameters of the spectral model used during analysis (LPC and gain parameters) are transmitted separately at the rate of 2.3 kb/s, leaving about 7.3 kb/s for the baseband residual.

In a first experiment, we implemented integer-band spectral translation in the DCT domain. We found the frequency-domain results to be perceptually similar to the time-domain results, with low-level tones and no roughness in the background. In a second experiment, we implemented the proposed pitch-adaptive HFR method and found that it largely eliminates the low-level tones, but it introduces a certain amount of roughness reminiscent of rectification. Finally, in a third experiment, we performed the pitch-adaptive HFR method with the added cross-correlation feature for better spectral duplication. Upon informal listening, we found that this system provides a marked improvement in speech quality over the traditional waveform rectification approach and over the non-pitch-adaptive time-domain spectral duplication methods. We expect the quality of the synthetic speech to improve further when we change the coding of the baseband from APCM to ATC.

Conclusions

In this paper we described an adaptive-transform baseband coder. The salient features of this coder are: (a) Transmitting the DCT of the baseband residual, and (b) regenerating the missing high frequency components in a pitch-adaptive manner. Since the transmitted information is in the frequency domain, the method lends itself very easily to the pitch-adaptive spectral translation method of high-frequency regeneration. Also, the method we described for transform coding the residual is applicable to the full-band residual, or any fraction thereof. Therefore, this coder is an attractive possibility as a multi-rate system.

Acknowledgment

This work was sponsored by the Advanced Research Projects Agency and monitored by RADC/ETC under contract number F19628-78-C-0136.

References

[1] M.R. Schroeder, "Recent Progress in Speech Coding at Bell Telephone Laboratories," in Proc. IIIrd Int. Cong. Acoust., Stuttgart, Germany, 1959.
[2] M.R. Schroeder and E.E. David, Jr., "A Vocoder for Transmitting 10 kc/s Speech Over a 3.5 kc/s Channel," Acustica, Vol. 10, pp. 35-43, 1960.
[3] E.E. David, M.R. Schroeder, B.F. Logan, and A.J. Prestigiacomo, "New Applications of Voice-Excitation of Vocoders," in Proc. Stockholm Speech Comm. Seminar, Royal Inst. Tech., Stockholm, Sweden, 1962.
[4] B. Gold and J. Tierney, "Digitized Voice-Excited Vocoder for Telephone-Quality Inputs, Using Bandpass Sampling of the Baseband Signals," J. Acoust. Soc. Amer., Vol. 37, pp. 753-754, April 1965.
[5] C.K. Un and D.T. Magill, "The Residual-Excited Linear Prediction Vocoder with Transmission Rate Below 9.6 kb/s," IEEE Trans. Comm., Vol. COM-23, pp. 1466-1474, Dec. 1975.
[6] C.J. Weinstein, "A Linear Prediction Vocoder with Voice Excitation," Proc. EASCON '75, pp. 30A-30G, Sept. 29-Oct. 1, 1975, Washington, D.C.
[7] D. Esteban, C. Galand, D. Mauduit, and J. Menez, "9.6/7.2 KBPS Voice Excited Predictive Coder (VEPC)," Int. Conf. Acoustics, Speech, and Signal Processing, Tulsa, OK, pp. 307-311, April 1978.
[8] B.S. Atal, M.R. Schroeder, and V. Stover, "Voice-Excited Predictive Coding System for Low Bit-Rate Transmission of Speech," Int. Conf. Comm., June 16-18, 1975, San Francisco.
[9] R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals," IEEE Trans. Acoustics, Speech, Signal Processing, Vol. ASSP-25, pp. 299-309, Aug. 1977.
[10] J.M. Tribolet and R.E. Crochiere, "A Vocoder-Driven Adaption Strategy for Low-Bit Rate Adaptive Transform Coding of Speech," Proc. 1978 Int. Conf. on Digital Signal Processing, Florence, Italy, Sept. 1978.
[11] J.M. Tribolet and R.E. Crochiere, "An Analysis/Synthesis Framework for Transform Coding of Speech," Int. Conf. Acoustics, Speech, and Signal Processing, Washington, D.C., pp. 81-84, April 1979.
[12] R. Viswanathan, W. Russell, and J. Makhoul, "Voice-Excited LPC Coders for 9.6 kbps Speech Transmission," Int. Conf. Acoustics, Speech, and Signal Processing, Washington, D.C., pp. 558-561, April 1979.
[13] J. Makhoul and M. Berouti, "High-Frequency Regeneration in Speech Coding Systems," Int. Conf. Acoustics, Speech, and Signal Processing, Washington, D.C., pp. 428-431, April 1979.

APPENDIX D

A FAST COSINE TRANSFORM IN ONE AND TWO DIMENSIONS

(IEEE Trans. Acoustics, Speech and Signal Processing, pp. 27-34, Aug. 1980)

A Fast Cosine Transform in One and Two Dimensions

JOHN MAKHOUL, SENIOR MEMBER, IEEE

Abstract—The discrete cosine transform (DCT) of an N-point real signal is derived by taking the discrete Fourier transform (DFT) of a $2N$-point even extension of the signal. It is shown that the same result may be obtained using only an N-point DFT of a reordered version of the original signal, with a resulting saving of 1/2. If the fast Fourier transform (FFT) is used to compute the DFT, the result is a fast cosine transform (FCT) that can be computed using on the order of $N \log_2 N$ real multiplications. The method is then extended to two dimensions, with a saving of 1/4 over the traditional method that uses the DFT.

I. Introduction

THE discrete cosine transform (DCT) has had a number of applications in image processing (see [1]) and, more recently, in speech processing [2], [3]. Compared to other orthogonal transforms, its performance seems to compare most favorably with the optimal Karhunen-Loève transform of a large number of signal classes [2], [4]. It has been shown that the DCT of an N-point signal can be computed using a $2N$-point discrete Fourier transform (DFT) [4]. Chen *et al.* [1] used matrix factorization to derive a special algorithm to compute the DCT of a signal with N a power 2, resulting in a saving of 1/2 over the previous method when the latter uses the fast Fourier transform (FFT).[1] More recently, Narasimha and Peterson [7] developed a method that employs an N-point DFT of a reordered version of the signal (where N is assumed to be even), resulting in a similar saving of 1/2. When N is a power of 2, use of the FFT results in a saving comparable to that of Chen *et al.* [1]. However, in the algorithm of Narasimha and Peterson, one can use existing software to compute the FFT instead of implementing a special algorithm for the DCT.

The algorithm presented here for the 1-D case is essentially identical to that of Narasimha and Peterson [7].[2] Our algorithm is more general in that N may be odd or even. This generalization and the extension to the 2-D case are facilitated by the view taken that the DCT can be regarded essentially as the DFT of an even extension of the signal. The generalization to the m-D case should then be straightforward, with a resulting saving of $1/2^m$ over the traditional method that employs the DFT. In [7] the authors mention that the inverse DCT can be obtained using a number of computations equal to that of the forward DCT. Here, we show how this can be done. Procedures for the forward and inverse fast cosine transforms are presented for easy implementation on the computer.

Finally, an appendix is included that presents a method and associated flowgraphs for efficient computation of the DFT of a real sequence and the IDFT of a Hermitian symmetric sequence. The flowgraphs are believed to be novel.

II. Discrete Cosine Transform

To motivate the derivation of the DCT presented below, we shall first review some basic discrete-time Fourier theory.

Let $x(n)$ be a discrete-time signal and $X(\omega)$ its Fourier transform. In one definition the cosine transform of $x(n)$ is the real part of $X(\omega)$. The real part of $X(\omega)$ is also equal to the Fourier transform of the even part of $x(n)$, defined by $x_e(n) = [x(n) + x(-n)]/2$ (see [8], for example). Therefore, the cosine transform of $x(n)$ is equal to the Fourier transform of $x_e(n)$. Now, if $x(n)$ is causal, i.e., $x(n) = 0$ for $n < 0$, then $x_e(n)$ and, therefore, the cosine transform uniquely specifies $x(n)$. In that case, $x_e(n)$ can be thought of as an even extension of $x(n)$. Therefore, the cosine transform of a causal $x(n)$ can be obtained as the Fourier transform of an even extension of $x(n)$. This viewpoint forms the basis of the derivation of the DCT below.

As an example, Fig. 1(a) shows a causal signal $x(n)$, and Fig. 1(b) shows an even extension of $x(n)$, $y_1(n) = x(n) + x(-n)$. [$y_1(n)$ is equal to twice the even part of $x(n)$.] Another possible even extension of $x(n)$ is shown in Fig. 1(c), where $y_2(n) = x(n) + x(-n-1)$. $y_1(n)$ is even about the point $n = 0$ while $y_2(n)$ is even about $n = -0.5$. The Fourier transform of $y_1(n)$ is real, while the Fourier transform of $y_2(n)$ contains a linear-phase term corresponding to the half-sample offset. Cosine transforms based on $y_1(n)$ or $y_2(n)$ can be defined and from which $x(n)$ can be determined uniquely.

In the example above we assumed that the Fourier transform is computed at all frequencies. In practice, one usually computes the *discrete* Fourier transform (DFT) at a finite number of equally spaced frequencies. For this case, the signal can be recovered in its aliased periodic form from the DFT [8]. In attempting to form even extensions of a signal where the extended signals are constrained to be periodic, one has addi-

Manuscript received November 27, 1978; revised April 25, 1979 and August 28, 1979. This work was supported by the Advanced Research Projects Agency and monitored by RADC/LTC under Contract F 19628-78-C-0136.

The author is with Bolt Beranek and Newman, Inc., Cambridge, MA 02138.

[1] Chen *et al.* [1] claim a larger saving. However, if in the conventional method one takes advantage of the fact that the signal is real, then the saving amounts to only 1/2.

[2] [7] was unknown to the author when this paper was first submitted for publication. The author thanks R. Crochiere and the reviewers for bringing [7] to his attention. The parts of this paper that overlap [7] have been retained to enhance the tutorial aspect of this paper.

0096-3518/80/0200-0027$00.75 © 1980 IEEE

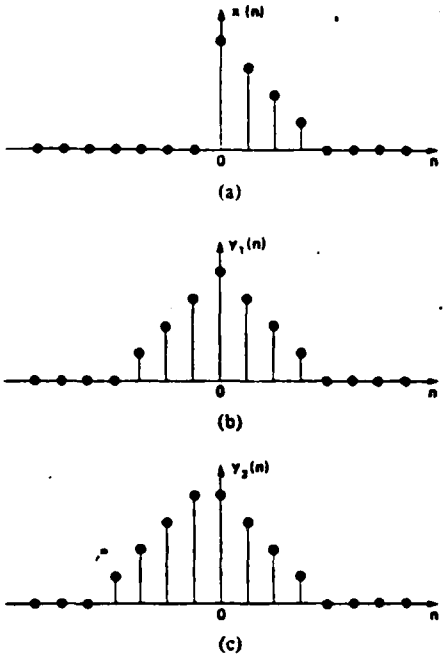


Fig. 1. (a) Causal signal $x(n)$. (b) An even extension of $x(n)$, $y_1(n)$, about $n = 0$. (c) Another even extension of $x(n)$, $y_2(n)$, about $n = -0.5$.

tional choices in defining those extensions. As an example, let $x(n)$, $0 \leq n \leq N-1$, be the sequence given by the four nonzero samples in Fig. 1(a) (i.e., $N = 4$). Fig. 2 shows four different even extensions of $x(n)$. $y_1(n)$ is a $(2N-2)$-point even extension; $y_2(n)$ and $y_3(n)$ are two different $(2N-1)$-point even extensions; and $y_4(n)$ is a $2N$-point even extension. Each of the four extension definitions could form the basis for a DCT definition. The most common form of the DCT is the one derived from the $2N$-point even extension $y_4(n)$ and is the one discussed in this paper.

A. Forward DCT

We desire the DCT of an N-point real data sequence $x(n)$, $0 \leq n \leq N-1$. The DCT is derived below from the DFT of a $2N$-point even extension of $x(n)$.

Let $y(n)$ be a $2N$-point even extension of $x(n)$ defined by:

$$y(n) = \begin{cases} x(n), & 0 \leq n \leq N-1 \\ x(2N-n-1), & N \leq n \leq 2N-1. \end{cases} \tag{1}$$

Then

$$y(2N-n-1) = y(n). \tag{2}$$

Fig. 3(a) shows an example of a signal $x(n)$, and Fig. 3(b) shows the corresponding even extension of $x(n)$ as defined in (1). Because of the minus 1 on the left-hand side of (2), $y(n)$ is not even about N and, therefore, will not have a real DFT, as we shall see below.

The DFT of $y(n)$ is given by

$$Y(k) = \sum_{n=0}^{2N-1} y(n)\, W_{2N}^{nk} \tag{3}$$

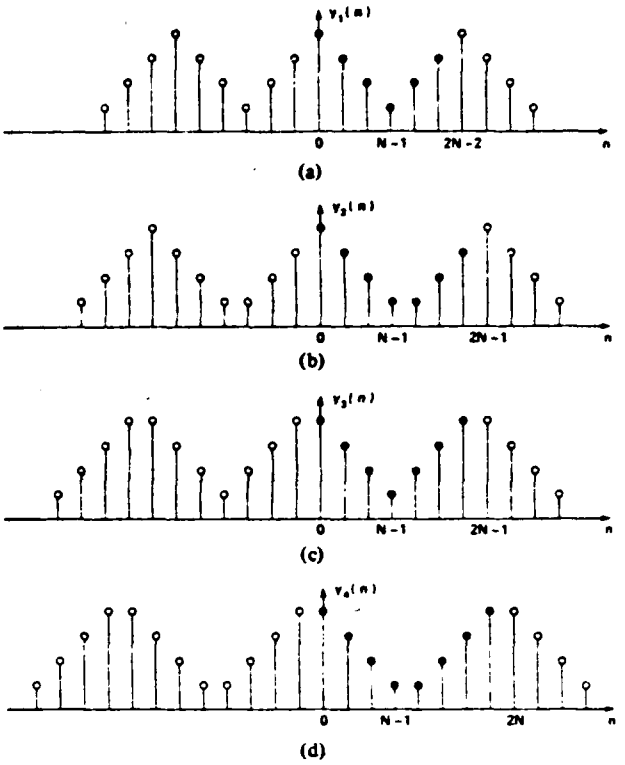


Fig. 2. Four different periodic even extensions of the nonzero $x(n)$ samples in Fig. 1(a). (a) $y_1(n)$ is a $(2N-2)$-point even extension. (b) and (c) $y_2(n)$ and $y_3(n)$ are two different $(2N-1)$-point even extensions. (d) $y_4(n)$ is a $2N$-point even extension. The DCT discussed in this paper is derived from the $2N$-point even extension $y_4(n)$.

where

$$W_M = e^{-j2\pi/M}. \tag{4}$$

Substituting (1) in (3), we have:

$$Y(k) = \sum_{n=0}^{N-1} x(n)\, W_{2N}^{nk} + \sum_{n=N}^{2N-1} x(2N-n-1)\, W_{2N}^{nk}.$$

By changing the summation variable in the right-hand term, noting that $W_{2N}^{2mN} = 1$ for integer m, and factoring out $W_{2N}^{-k/2}$, we have

$$Y(k) = W_{2N}^{-k/2} \sum_{n=0}^{N-1} x(n) \left[W_{2N}^{nk} W_{2N}^{k/2} + W_{2N}^{-nk} W_{2N}^{-k/2} \right]. \tag{5}$$

The expression in (5) may be written in two ways:

$$Y(k) = W_{2N}^{-k/2} \cdot 2 \sum_{n=0}^{N-1} x(n) \cos \frac{\pi(2n+1)k}{2N}, \qquad 0 \leq k \leq 2N-1 \tag{6}$$

or

$$Y(k) = W_{2N}^{-k/2} \cdot 2 \operatorname{Re} \left[W_{2N}^{k/2} \sum_{n=0}^{N-1} x(n)\, W_{2N}^{nk} \right], \qquad 0 \leq k \leq 2N-1. \tag{7}$$

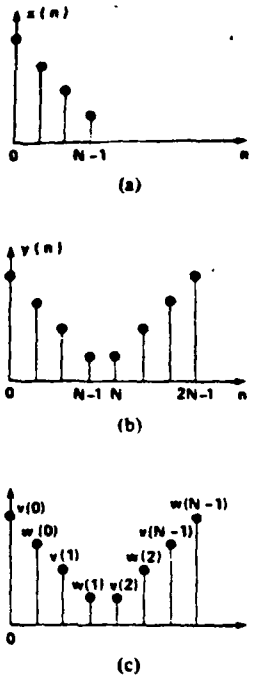


Fig. 3. (a) Original signal $x(n)$, $0 \leq n \leq N-1$. (b) A $2N$-point even extension of $x(n)$, $y(n)$. (c) Division of $y(n)$ into its even and odd parts $v(n)$ and $w(n)$.

By defining the DCT of $x(n)$ as [3]

$$C(k) = 2\sum_{n=0}^{N-1} x(n)\cos\frac{\pi(2n+1)k}{2N}, \quad 0 \leq k \leq N-1, \tag{8}$$

we have, from (6) and (8),

$$Y(k) = W_{2N}^{-k/2}\, C(k) \tag{9a}$$

or

$$C(k) = W_{2N}^{k/2}\, Y(k) \tag{9b}$$

and, from (7) and (9a),

$$C(k) = 2\,\mathrm{Re}\left[W_{2N}^{k/2}\sum_{n=0}^{N-1} x(n)\, W_{2N}^{nk}\right]. \tag{10}$$

Equation (9) specifies the relationship between the DCT of a sequence and the DFT of the $2N$-point extension of that sequence. Note that $C(k)$ is real and $Y(k)$ is complex. $Y(k)$ would have been equal to $C(k)$ had the sequence $y(n)$ been delayed by half a sample, in which case $y(n)$ would have been truly even.

Therefore, the DCT of $x(n)$ may be computed by taking the $2N$-point DFT of $y(n)$, as in (3), and multiplying the result by $W_{2N}^{k/2}$, as in (9b). From (10) we see that the DCT may also be obtained by taking the $2N$-point DFT of the original sequence $x(n)$ with N zeros appended to it, multiplying the result by $W_{2N}^{k/2}$, then taking twice the real part. The latter method has been the one in common usage [4].

[3]The DCT definition here is slightly different from other definitions [4], mainly in the relative amplitude of $C(0)$ to that of other terms; also, we do not use an orthonormalizing factor. The range on k is the same here as in the literature; however, in this paper we shall also make use of $C(N)$, which, from (8), is equal to zero always since the cosine term is zero for $k = N$.

B. Inverse DCT

Again, we shall derive the inverse DCT (IDCT) from the inverse DFT (IDFT). The IDFT of $Y(k)$ is given by

$$y(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} Y(k)\, W_{2N}^{-nk}. \tag{11}$$

Since $y(n)$ is real, $Y(k)$ is Hermitian symmetric:

$$Y(2N-k) = Y^*(k). \tag{12}$$

Furthermore, from (7), it is simple to show that

$$Y(N) = 0. \tag{13}$$

Using (12) and (13) in (11), one can show that

$$y(n) = \frac{1}{N}\,\mathrm{Re}\left[\frac{Y(0)}{2} + \sum_{k=1}^{N-1} Y(k)\, W_{2N}^{-nk}\right],$$

$$0 \leq n \leq 2N-1. \tag{14}$$

Substituting (9a) in (14) and using (1), we have the desired IDCT

$$x(n) = \frac{1}{N}\left[\frac{C(0)}{2} + \sum_{k=1}^{N-1} C(k)\cos\frac{\pi(2n+1)k}{2N}\right],$$

$$0 \leq n \leq N-1. \tag{15}$$

Equations (8) and (15) form a DCT pair. Given $C(k)$, $x(n)$ is retrieved by first computing $Y(k)$ using (9a), then taking the $2N$-point *complex* IDFT implied by (14), which results in $y(n)$ and, hence, $x(n)$.

III. Fast Cosine Transform (FCT)

A. Forward FCT

We now show that the DCT may be obtained from the N-point DFT of a real sequence instead of a $2N$-point DFT, resulting in a saving of 1/2.

Divide the sequence $y(n)$ into two N-point sequences

$$\left.\begin{aligned} v(n) &= y(2n) \\ w(n) &= y(2n+1) \end{aligned}\right\} \quad 0 \leq n \leq N-1 \tag{16}$$

where $v(n)$ and $w(n)$ are the sets of even and odd points in $y(n)$, respectively. Fig. 3(c) shows how this division takes place in the given example. Note that each of $v(n)$ and $w(n)$ contain *all* the original samples of $x(n)$, and that $w(n)$ is simply the reverse sequence of $v(n)$. In fact, from (2) and (16), one can show that

$$w(n) = v(N-n-1), \quad 0 \leq n \leq N-1. \tag{17}$$

Substituting (16) in (3), we have

$$Y(k) = \sum_{n=0}^{N-1} v(n)\, W_{2N}^{2nk} + \sum_{n=0}^{N-1} w(n)\, W_{2N}^{(2n+1)k}. \tag{18}$$

Substituting (17) in (18), noting that $W_{2N}^{2nk} = W_N^{nk}$, rearranging terms, and using (9), one can show that

$$C(k) = 2\,\mathrm{Re}\left[W_{4N}^{k}\sum_{n=0}^{N-1} v(n)\,W_N^{nk}\right], \quad 0 \leq k \leq N-1. \tag{19a}$$

The difference between (19a) and (10) is the use of W_N instead of W_{2N} in the summation, and $v(n)$ instead of $x(n)$. The result is that one can now compute the DCT from an N-point DFT instead of a $2N$-point DFT. Equation (19a) may be rewritten as

$$C(k) = 2\sum_{n=0}^{N-1} v(n)\cos\frac{\pi(4n+1)k}{2N}, \quad 0 \leq k \leq N-1, \tag{19b}$$

which gives an alternate definition of the DCT in terms of the reordered sequence $v(n)$ [compare (19b) with (8)].

The sequence $v(n)$ can be written directly in terms of $x(n)$:

$$v(n) = \begin{cases} x(2n), & 0 \leq n \leq \left[\dfrac{N-1}{2}\right] \\ x(2N-2n-1), & \left[\dfrac{N+1}{2}\right] \leq n \leq N-1 \end{cases} \tag{20}$$

where $[a]$ denotes "integer part of a." Therefore, $v(n)$ is obtained by taking the even points in $x(n)$ in order, followed by the odd points in their reverse order. Note that (20) applies for any value of N, odd or even.

The specification of the FCT is now complete; the computational procedure is given in Section III-B. We simply note here that since $v(n)$ is real, its N-point DFT can be computed from the $(N/2)$-point DFT of a complex sequence (see the Appendix).

B. Inverse FCT

The idea here is to compute $v(n)$ from the DCT first, then use (20) to obtain $x(n)$. Substituting $v(n) = y(2n)$ in (14), we have

$$v(n) = \frac{1}{N}\,\mathrm{Re}\left[\frac{Y(0)}{2} + \sum_{k=1}^{N-1} Y(k)\,W_N^{-nk}\right], \quad 0 \leq n \leq N-1. \tag{21}$$

Equation (21) indicates that $v(n)$ can be computed using an N-point complex IDFT instead of the $2N$-point complex IDFT implied by (14). However, the number of computations is still about twice that used in the forward FCT. We now show that the inverse FCT can, in fact, be computed with the same number of computations as the forward FCT. The method is to compute $V(k)$ from $C(k)$, then compute the IDFT of $V(k)$ to obtain $v(n)$.

Equation (19a) can be rewritten as

$$C(k) = \mathrm{Re}\left[2W_{4N}^{k}\,V(k)\right] \tag{22}$$

where $V(k)$ is the DFT of $v(n)$. To compute $V(k)$ from $C(k)$ in (22) we need also a knowledge of the imaginary part of the term in brackets. Denote the imaginary part by $C_i(k)$ and the whole complex number by $C_c(k)$, where

$$C_c(k) = C(k) + jC_i(k) = 2W_{4N}^{k}\,V(k); \tag{23}$$

then

$$V(k) = \tfrac{1}{2}\,W_{4N}^{-k}\,C_c(k). \tag{24}$$

We first need to determine $C_i(k)$. Using the fact that $V(k)$ is Hermitian symmetric

$$V(N-k) = V^*(k), \tag{25}$$

one can show, using (23), that

$$C_c(N-k) = -jC_c^*(k) = -[C_i(k) + jC(k)]. \tag{26}$$

From (23) and (26), we conclude that

$$C_i(k) = -C(N-k)$$

and

$$C_c(k) = C(k) - jC(N-k) = 2W_{4N}^{k}\,V(k). \tag{27}$$

(Note that one can take advantage of (27) in (23) for computing $C(k)$ since one can compute $C(k)$ and $C(N-k)$ simultaneously.) From (27), we have

$$V(k) = \tfrac{1}{2}\,W_{4N}^{-k}[C(k) - jC(N-k)], \quad 0 \leq k \leq N-1. \tag{28}$$

$V(k)$ is computed from (28) for $0 \leq k \leq N/2$, then use (25) for $k > N/2$. In computing $V(0)$, one needs the value of $C(N)$, which, from (9b) and (13), is seen easily to be equal to zero

$$C(N) = 0. \tag{29}$$

After computing $V(k)$, $v(n)$ is obtained as the IDFT

$$v(n) = \frac{1}{N}\sum_{k=0}^{N-1} V(k)\,W_N^{-nk}. \tag{30}$$

It would seem that (30) again requires an N-point complex IDFT. However, in the Appendix we show how the IDFT of a Hermitian symmetric sequence can be computed using an $(N/2)$-point complex IDFT, the same as in the forward FCT.

We are now ready to specify the complete procedure for computing the FCT and the inverse FCT.

FCT Procedure: Given a real sequence $x(n)$, $0 \leq n \leq N-1$.

1) Form the sequence $v(n)$ from (20).

2) Compute $V(k)$, $0 \leq k \leq N-1$, the DFT of $v(n)$.

3) Multiply $V(k)$ by $2\exp(-j\pi k/2N)$. From (27), we see that the real part will be $C(k)$ and the negative of the imaginary part will be $C(N-k)$. Therefore, the value of k is varied in the range $0 \leq k \leq [N/2]$.

IFCT Procedure. Given the DCT $C(k)$, $0 \leq k \leq N-1$, and $C(N) = 0$.

1) Compute $V(k)$ from (28).

2) Compute the IDFT of $V(k)$, $v(n)$.

3) Retrieve $x(n)$ from $v(n)$ using (20).

C. Computational Considerations

Since most researchers have some form of the DFT available on their computers, the FCT and IFCT procedures given in

the previous section can be implemented easily. Furthermore, the procedure is quite general and may be used for any value of N. We have also seen that the N-point DFT of $v(n)$ and the N-point IDFT of $V(k)$ can be computed from the $(N/2)$-point DFT and $(N/2)$-point IDFT, respectively, of some complex sequence. For a highly composite value of N, one can, of course, use the FFT to great advantage. Maximum savings accrue when N is a power of 2. In the latter case, if one makes use of the fact that the sequence is real, the total number of computations for the FCT or the IFCT is on the order of $N \log_2 N$, the same as in [1]. The major difference here is that we do not require a specialized algorithm.

In computing $C(k)$ from (27) one first takes the N-point DFT of $v(n)$, and therefore one needs a table of sines or cosines where the unit circle is divided into N equal segments. However, multiplying afterwards by W_{4N}^{k} requires a table where the unit circle is divided into $4N$ segments. Since k is in the range $0 \leq k \leq N-1$, the N values of sines and cosines are all in the first quadrant. This point is made to emphasize the fact that the DCT of a sequence of length N requires an exponential table four times as large as that required for an N-point DFT.

IV. Two-Dimensional Fast Cosine Transform

In this section we present results analogous to the 1-D case given in Sections II and III. We show how the DCT of a 2-D real sequence $\{x(n_1, n_2), 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1\}$ can be computed using an $(N_1 \times N_2)$-point real DFT instead of the $(2N_1 \times 2N_2)$-point real DFT required in the traditional method, resulting in a saving of 1/4. Since the methods used here are similar to the 1-D case, no detailed derivations will be given.

A. Two-Dimensional DCT and IDCT

In a manner analogous to the 1-D case in (1), define a $(2N_1 \times 2N_2)$-point even extension of $x(n_1, n_2)$ in the n_1 and n_2 directions:

$$y(n_1, n_2) = \begin{cases} x(n_1, n_2); & 0 \leq n_1 \leq N_1 - 1, \quad 0 \leq n_2 \leq N_2 - 1 \\ y(2N_1 - n_1 - 1, n_2); & N_1 \leq n_1 \leq 2N_1 - 1, \quad 0 \leq n_2 \leq N_2 - 1 \\ y(n_1, 2N_2 - n_2 - 1); & 0 \leq n_1 \leq N_1 - 1, \quad N_2 \leq n_2 \leq 2N_2 - 1 \\ y(2N_1 - n_1 - 1, 2N_2 - n_2 - 1); & N_1 \leq n_1 \leq 2N_1 - 1, \quad N_2 \leq n_2 \leq 2N_2 - 1. \end{cases} \tag{31}$$

Fig. 4 shows an example where $N_1 = 3$ and $N_2 = 4$; the numbers in the figure are the sample values. Note that the number of samples in $y(n_1, n_2)$ is $4N_1N_2$, i.e., four times that in $x(n_1, n_2)$.[4] The 2-D DFT of $y(n_1, n_2)$ is defined by

$$Y(k_1, k_2) = \sum_{n_1=0}^{2N_1-1} \sum_{n_2=0}^{2N_2-1} y(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}. \tag{32}$$

From (31) and (32), one can show that

$$Y(k_1, k_2) = W_{2N_1}^{-k_1/2} W_{2N_2}^{-k_2/2} C(k_1, k_2) \tag{33}$$

[4] In the m-D case, the number of samples in the extended sequence $y(n_1, \cdots, n_m)$ is 2^m times the number of samples in $x(n_1, \cdots, n_m)$.

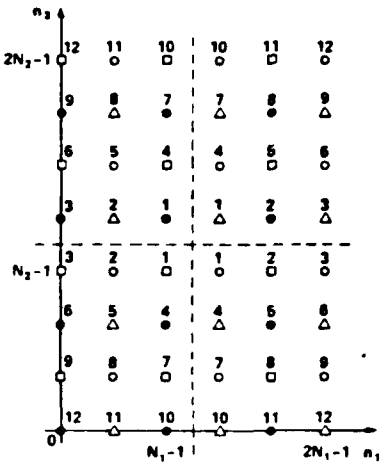


Fig. 4. An example of a 2-D $(N_1 \times N_2)$-point sequence and its $(2N_1 \times 2N_2)$-point even extension. Here, $N_1 = 3$ and $N_2 = 4$. The sample values are given numerically in the figure. The four sequences defined in (39) in the text are indicated in the figure as follows: filled circles, $v(n_1, n_2)$; triangles, $w_1(n_1, n_2)$; squares, $w_2(n_1, n_2)$; open circles, $w_3(n_1, n_2)$.

where

$$C(k_1, k_2) = 4 \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cos \frac{\pi(2n_1 + 1)k_1}{2N_1} \cdot \cos \frac{\pi(2n_2 + 1)k_2}{2N_2}, \quad 0 \leq k_1 \leq N_1 - 1, \quad 0 \leq k_2 \leq N_2 - 1 \tag{34}$$

is the 2-D DCT of the sequence $x(n_1, n_2)$. The computation of $C(k_1, k_2)$ from either (32) and (33) or from (34) requires one to take the DFT of a $(2N_1 \times 2N_2)$-point real sequence.

From (32)-(34), one can show that $Y(k_1, k_2)$ has the following properties:

$$\begin{aligned} Y(2N_1 - k_1, 2N_2 - k_2) &= Y^*(k_1, k_2) \\ Y(2N_1 - k_1, 0) &= Y^*(k_1, 0) \\ Y(0, 2N_2 - k_2) &= Y^*(0, k_2) \\ Y(N_1, k_2) &= Y(k_1, N_2) = 0. \end{aligned} \tag{35}$$

The first three equations constitute the Hermitian symmetric properties in 2-D, and they are derivable from (32) for real $y(n_1, n_2)$. The last equation in (35) is analogous to (13) in 1-D, and is a consequence of the particular type of even symmetry of $y(n_1, n_2)$. Substituting (35) in the equation for the 2-D IDFT

$$y(n_1, n_2) = \frac{1}{4N_1N_2} \sum_{k_1=0}^{2N_1-1} \sum_{k_2=0}^{2N_2-1} Y(k_1, k_2)\, W_{2N_1}^{-n_1k_1} W_{2N_2}^{-n_2k_2}, \tag{36}$$

one can show that the 2-D IDCT of $C(k_1, k_2)$ is given by

$$x(n_1, n_2) = \frac{1}{N_1N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_1=0}^{N_2-1} C'(k_1, k_2) \cdot \cos \frac{\pi(2n_1+1)k_1}{2N_1} \cos \frac{\pi(2n_2+1)k_2}{2N_2} \tag{37}$$

where

$$C'(k_1, k_2) = \begin{cases} C(0,0)/4, & k_1 = 0,\ k_2 = 0 \\ C(k_1,0)/2, & k_1 \neq 0,\ k_2 = 0 \\ C(0,k_2)/2, & k_1 = 0,\ k_2 \neq 0 \\ C(k_1,k_2), & k_1 \neq 0,\ k_2 \neq 0. \end{cases} \tag{38}$$

$x(n_1, n_2)$ may be computed using a 2-D $(2N_1 \times 2N_2)$-point IDFT.

B. Two-Dimensional FCT and IFCT

The corresponding fast algorithms in the 2-D case are obtained in a manner analogous to the 1-D case. We divide $y(n_1, n_2)$ into four $(N_1 \times N_2)$-point sequences, starting with the four points at (0, 0), (1, 0), (0, 1), and (1, 1), respectively, and taking every second point after that.[5] The four sequences are then given by

$$\begin{aligned} v(n_1, n_2) &= y(2n_1, 2n_2) \\ w_1(n_1, n_2) &= y(2n_1+1, 2n_2) \\ w_2(n_1, n_2) &= y(2n_1, 2n_2+1) \\ w_3(n_1, n_2) &= y(2n_1+1, 2n_2+1) \end{aligned} \tag{39}$$

$$v(n_1, n_2) = \begin{cases} x(2n_1, 2n_2); & 0 \leq n_1 \leq \left[\frac{N_1-1}{2}\right], \quad 0 \leq n_2 \leq \left[\frac{N_2-1}{2}\right] \\ x(2N_1-2n_1-1, 2n_2); & \left[\frac{N_1+1}{2}\right] \leq n_1 \leq N_1-1, \quad 0 \leq n_2 \leq \left[\frac{N_2-1}{2}\right] \\ x(2n_1, 2N_2-2n_2-1); & 0 \leq n_1 \leq \left[\frac{N_1-1}{2}\right], \quad \left[\frac{N_2+1}{2}\right] \leq n_2 \leq N_2-1 \\ x(2N_1-2n_1-1, 2N_2-2n_2-1); & \left[\frac{N_1+1}{2}\right] \leq n_1 \leq N_1-1, \quad \left[\frac{N_2+1}{2}\right] \leq n_2 \leq N_2-1. \end{cases} \tag{44}$$

where $0 \leq n_1 \leq N_1 - 1$ and $0 \leq n_2 \leq N_2 - 1$. In the example in Fig. 4 note how each of the sequences in (39) contains all of the samples in $x(n_1, n_2)$, but in a reordered fashion. From (39) and (31), one can show that

$$\begin{aligned} w_1(n_1, n_2) &= v(N_1 - n_1 - 1, n_2) \\ w_2(n_1, n_2) &= v(n_1, N_2 - n_2 - 1) \\ w_3(n_1, n_2) &= v(N_1 - n_1 - 1, N_2 - n_2 - 1). \end{aligned} \tag{40}$$

Let $V(k_1, k_2)$ be the 2-D $(N_1 \times N_2)$-point DFT of $v(n_1, n_2)$:

$$V(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} v(n_1, n_2)\, W_{N_1}^{n_1k_1} W_{N_2}^{n_2k_2}. \tag{41}$$

Then, by substituting (40) and (39) in (32), and using (33), one can show that

$$C(k_1, k_2) = 2\,\mathrm{Re}\{W_{4N_1}^{k_1}[W_{4N_2}^{k_2} V(k_1, k_2) + W_{4N_2}^{-k_2} V(k_1, N_2 - k_2)]\} \tag{42a}$$

or equivalently,

$$C(k_1, k_2) = 2\,\mathrm{Re}\{W_{4N_2}^{k_2}[W_{4N_1}^{k_1} V(k_1, k_2) + W_{4N_1}^{-k_1} V(N_1 - k_1, k_2)]\}. \tag{42b}$$

From (42), it is simple to show that

$$C(k_1, k_2) = 4 \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} v(n_1, n_2) \cos \frac{\pi(4n_1+1)k_1}{2N_1} \cdot \cos \frac{\pi(4n_2+1)k_2}{2N_2}. \tag{43}$$

It is clear from (42) and (41) that $C(k_1, k_2)$ can be computed using an $(N_1 \times N_2)$-point DFT instead of a $(2N_1 \times 2N_2)$-point DFT, resulting in a saving of 1/4.

The only remaining step is how to obtain $v(n_1, n_2)$ directly from $x(n_1, n_2)$ instead of using (39). One can show that

The 2-D IFCT of $C(k_1, k_2)$ is obtained by first computing $V(k_1, k_2)$ from $C(k_1, k_2)$. In a manner analogous to the 1-D case, we define a complex quantity $C_c(k_1, k_2)$ by not taking the real part in (42a), and then show that

$$C_c(k_1, k_2) = C(k_1, k_2) - jC(N_1 - k_1, k_2). \tag{45}$$

Find $C_c^*(N_1 - k_1, N_2 - k_2)$, then add and subtract the result from $C_c(k_1, k_2)$. The answer can be shown to reduce to

$$V(k_1, k_2) = \tfrac{1}{4} W_{4N_1}^{-k_1} W_{4N_2}^{-k_2} \{[C(k_1, k_2)$$

[5] In the m-D case, $y(n_1, \cdots, n_m)$ is divided into 2^m sequences, starting with each of the 2^m corners of the unit m-D cube, from $(0, 0, \cdots, 0)$ to $(1, 1, \cdots, 1)$, and taking every second point after that. The sequence that begins at $(i_1, i_2, \cdots, i_m)$, where each $i_k = 0$ or 1, is defined by $y(2n_1 + i_1, 2n_2 + i_2, \cdots, 2n_m + i_m)$.

$$- C(N_1 - k_1, N_2 - k_2)] - j[C(N_1 - k_1, k_2) + C(k_1, N_2 - k_2)]\} \quad (46)$$

where $0 \leq k_1 \leq N_1 - 1$ and $0 \leq k_2 \leq N_2 - 1$. However, since $V(k_1, k_2)$ is Hermitian symmetric, one need compute only half the values in (46). In performing the computations, one needs the fact that

$$C(N_1, k_2) = C(k_1, N_2) = 0, \quad \text{all } k_1 \text{ and } k_2, \quad (47)$$

which can be shown to be true from (33) and (35), or from (43).

After $V(k_1, k_2)$ is computed from (46) and (47), $v(n_1, n_2)$ is evaluated from the IDFT

$$v(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} V(k_1, k_2) W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2}. \quad (48)$$

The sequence $x(n_1, n_2)$ is then retrieved from $v(n_1, n_2)$ by using (44).

V. Conclusion

We showed how the DCT of an N-point sequence may be derived from the DFT of a $2N$-point even extension of the given sequence. Then, we presented a fast algorithm (FCT), first developed by Narasimha and Peterson [7], which allows for the computation of the DCT of a sequence from just an N-point DFT of a reordered version of the same sequence, with a resulting saving of 1/2. Therefore, one can use existing FFT software to compute the DCT. For N a power of 2, the number of computations is comparable to that reported by Chen *et al.* [1], who used a specialized algorithm.

The FCT algorithm in this paper is more general than that of [7] in that N may be odd or even. Also, an algorithm was developed here for computing the inverse FCT using the same number of computations as in the forward method.

The method was then extended to the 2-D case, where a saving of 1/4 was achieved. The method can be generalized to compute an m-D DCT using an m-D DFT, with a saving of $1/2^m$ over traditional methods that use the DFT.

Appendix
DFT and IDFT for a Real Sequence

Let $v(n)$, $0 \leq n \leq N - 1$ be a *real* sequence, where N is divisible by 2. We wish to compute the DFT of $v(n)$, $V(k)$, $0 \leq k \leq N - 1$, using an $N/2$-point DFT. Except for the flowgraphs in Fig. 5, the procedure given below is well known (see, for example, [6]).

DFT Procedure

1) Place the even and odd points of $v(n)$ in the real and imaginary parts, respectively, of a complex vector $t(n) = t_R(n) + jt_I(n)$, where

$$\left.\begin{aligned} t_R(n) &= v(2n) \\ t_I(n) &= v(2n+1) \end{aligned}\right\} 0 \leq n \leq \frac{N}{2} - 1. \quad (A\text{-}1)$$

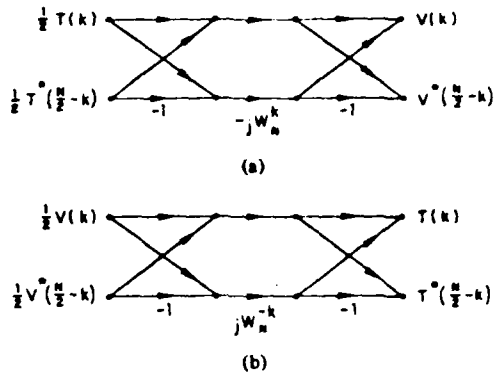


Fig. 5. (a) Supplementary flowgraph for computing the FFT of a real sequence. The computation in the figure is performed for $0 \leq k \leq [N/4]$. (b) Supplementary flowgraph for computing the IFFT of a Hermitian symmetric sequence. The computation is performed for $0 \leq k \leq [N/4]$.

2) Compute the $N/2$-point DFT of $t(n)$, $T(k)$, $0 \leq k \leq N/2 - 1$.

3) Compute $V(k)$ from $T(k)$ using the formula [5]

$$V(k) = \frac{1}{2}\left[T(k) + T^*\left(\frac{N}{2} - k\right)\right] - jW_N^k \frac{1}{2}\left[T(k) - T^*\left(\frac{N}{2} - k\right)\right]. \quad (A\text{-}2)$$

The computations in the last step can be made more efficient. From (A-2), one can write

$$V^*\left(\frac{N}{2} - k\right) = \frac{1}{2}\left[T(k) + T^*\left(\frac{N}{2} - k\right)\right] + jW_N^k \frac{1}{2}\left[T(k) - T^*\left(\frac{N}{2} - k\right)\right]. \quad (A\text{-}3)$$

Given $T(k)$, Fig. 5(a) shows the flowgraph [5] that implements (A-2) and (A-3) to compute $V(k)$. Note that the values of $V(k)$ are computed two at a time. Therefore, the value of k in Fig. 5(a) should range between $0 \leq k \leq [N/4]$. The other values of $V(k)$, $k > N/2$, may be obtained by noting that $V(k)$ is Hermitian symmetric. There are two points in $V(k)$ that are real and require no multiplication. They are

$$V(0) = \text{Re}\,[T(0)] + \text{Im}\,[T(0)]$$
$$V\left(\frac{N}{2}\right) = \text{Re}\,[T(0)] - \text{Im}\,[T(0)]. \quad (A\text{-}4)$$

Also, if N is divisible by 4, one can show that

$$V\left(\frac{N}{4}\right) = T^*\left(\frac{N}{4}\right).$$

Given a Hermitian symmetric $V(k)$, $0 \leq k \leq N - 1$, we wish to compute the IDFT, $v(n)$, $0 \leq n \leq N - 1$. From (A-2) and (A-3), one can easily solve for $T(k)$ and $T^*(N/2 - k)$. The resulting equations can be implemented using the flowgraph in Fig. 5(b). The IDFT procedure is, then, as shown in the following.

IDFT Procedure

1) Compute $T(k)$ from $V(k)$ using the flowgraph in Fig. 5(b), where the range of k in the figure is $0 \leq k \leq [N/4]$.

2) Compute the $(N/2)$-point IDFT of $T(k)$, $t(n)$, $0 \leq n \leq (N/2) - 1$.

3) $v(n)$ is obtained from $t(n)$ by using (A-1).

Finally, if N is not divisible by 2, one can compute the DFT of two separate sequences using the same method given above [6].

Acknowledgment

The author wishes to thank M. Berouti and the reviewers for their comments.

References

[1] W. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004-1009, Sept. 1977.

[2] R. Zelinski and P. Noll, "Adaptive transform coding of speech signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 299-309, Aug. 1977.

[3] J. M. Tribolet and R. E. Crochiere, "A vocoder-driven adaptation strategy for low-bit rate adaptive transform coding of speech," in *Proc. 1978 Int. Conf. Digital Signal Processing*, Florence, Italy, Aug. 30-Sept. 2, 1978.

[4] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-25, pp. 90-93, Jan. 1974.

[5] J. Makhoul, "Speaker-machine interaction in automatic speech recognition," Res. Lab. Electron., Massachusetts Inst. Tech., Cambridge, MA, Tech. Rep. 480, Dec. 15, 1970; also, Ph.D. dissertation, Dept. Elec. Eng., Massachusetts Inst. Tech., Cambridge, MA, 1970.

[6] E. O. Brigham, *The Fast Fourier Transform*. Englewood Cliffs, NJ: Prentice-Hall, 1974, pp. 166-169.

[7] M. J. Narasimha and A. M. Peterson, "On the computation of the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-26, pp. 934-936, June 1978.

[8] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

John Makhoul (S'64-M'70-SM'78) was born in Deirmimas, Lebanon, on September 19, 1942. He received the B.E. degree from the American University of Beirut, Beirut, Lebanon, in 1964, the M.Sc. degree from Ohio State University, Columbus, in 1965, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 1970, all in electrical engineering.

From 1964-1966 he did research in electromagnetic diffraction at the Antenna Laboratory, Ohio State University. From 1966-1970 he was a Research Assistant at the Research Laboratory of Electronics, Massachusetts Institute of Technology, working on speech synthesis, automatic speech recognition, and analog and digital systems design. Since 1970 he has been with Bolt Beranek and Newman, Inc., Cambridge, MA, where he is a Supervisory Scientist working primarily on speech analysis, synthesis and compression, speech enhancement, automatic speech recognition, and other aspects of digital signal processing, including lattice structures and adaptive digital filtering.

Dr. Makhoul is a Fellow of the Acoustical Society of America.

APPENDIX E

AN EMBEDDED-CODE MULTIRATE SPEECH TRANSFORM CODER

(IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, pp. 356-359, April 1980)

IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Denver, CO, April 1980

AN EMBEDDED-CODE MULTIRATE SPEECH TRANSFORM CODER

Michael Berouti and John Makhoul

Bolt Beranek and Newman Inc.
Cambridge, MA

ABSTRACT

In embedded multirate speech coding, it is desired to have a transmitter/receiver system that operates efficiently over a wide range of channel transmission rates. We are currently investigating a system based on adaptive transform coding of speech, in which we code and transmit the system parameters and the discrete cosine transform (DCT) coefficients of the fullband linear prediction residual waveform. The multirate property of the system is achieved by allowing the channel to discard some of the bits generated by the high data rate transmitter. Stripping off bits results in an absence of DCT components, which the receiver regenerates by a spectral duplication method. An inverse DCT at the receiver yields the time domain residual waveform to be used as input to the linear prediction synthesis filter. The lowest data rate achievable by the system is about 2.5 kb/s, in which case the system reduces to a narrowband LPC pitch-excited vocoder.

1. INTRODUCTION

In multirate speech coding schemes, it is desired to have a transmitter/receiver system that can operate over a wide range of channel transmission rates. With the added constraint of embedded coding, the channel is allowed to discard some of the transmitted bits at each frame to achieve lower data rates. Such a system lends itself well to a packet-switched communication network (such as the ARPANET), where traffic congestion would be alleviated by lowering the data rate.

In this paper we describe a hybrid linear-prediction transform-coding system that can operate in the range 2500-16000 b/s, with inputs bandlimited to 3.33 kHz and sampled at 6.67 kHz.

2. SYSTEM DESCRIPTION

Taking linear prediction (LP) as the basic method of spectral envelope representation, we code and transmit (i) the system parameters (LPC coefficients, gain, pitch, and pitch coefficient), and (ii) the discrete cosine transform (DCT) coefficients of the LP residual, using a modified adaptive transform coding (ATC) scheme [1]. A block diagram of the system is shown in Fig. 1.

At each frame, the codes representing the system parameters are transmitted first. These

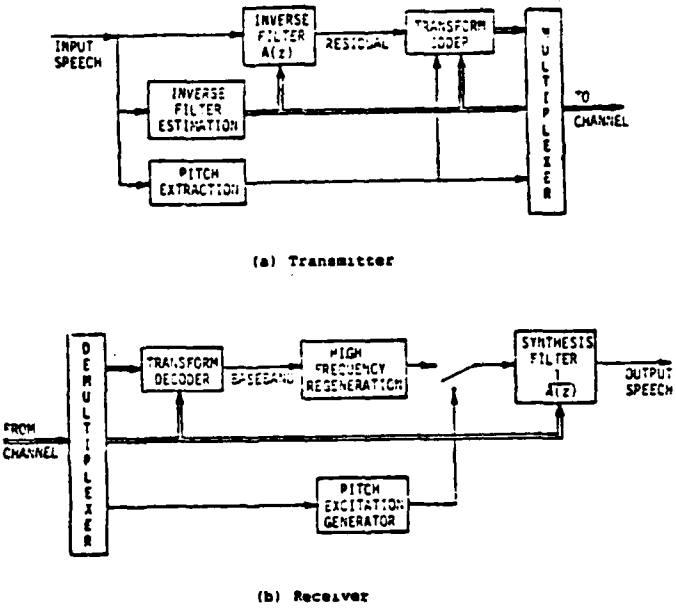


Fig. 1 Block diagram of multirate speech transform coder.

codes are then followed by the codes representing the DCT components. The transmitter always operates at the same fixed bit rate; it is the channel's function to discard some of the transmitted bits to alleviate traffic congestion. Thus, the maximum data rate of 16,000 b/s is achieved when the DCT of the fullband residual is transmitted. The minimum data rate achievable by the system takes place when all the codes representing the DCT components are suppressed by the channel and only the system parameters are transmitted. In such a case, the receiver becomes identical to that of a narrowband pitch-excited LPC vocoder operating at 2500 b/s.

Intermediate data rates are achieved by stripping off bits from the high data rate system. Stripping off bits results in the suppression of low-amplitude frequency components and/or high-frequency components. This aspect of the system will be explained in Section 4. Here, we point out that, at the intermediate data rates, the receiver regenerates the missing frequency components to restore the fullband DCT. An inverse DCT yields the time-domain residual waveform which

is used as input to the all-pole LP synthesis filter.

It is worth mentioning here that the transmitter itself can strip off bits prior to transmission in the same manner as is done by the channel. Thus, when a system is first turned on, the initial bit rate need not be high and traffic congestion can be avoided. Note that the receiver does not need to know where in the system the bits were discarded.

One salient feature of the above described system is that it is a frequency-domain baseband coder. It has the flexibility that the width of the baseband can vary in time, depending on the number of bits retained by the channel at each frame, thus accommodating various bit rates without affecting the analysis done at the transmitter and without the use of lowpass filters. Also, the advantage of transmitting the DCT of the residual is that the latter has a flat spectral envelope; it lends itself well to our previously developed methods of high-frequency regeneration by spectral duplication [2]. Once a flat-spectrum excitation signal is derived from the received baseband, the synthesis filter yields an output spectrum that is close to the spectrum of the input signal.

The LP synthesis filter also helps smooth the frame-boundary discontinuities caused by frequency-domain quantization (time-domain aliasing). In fact, unlike ordinary ATC schemes, we have found that no overlap between frames is necessary when we use the DCT of the residual.

Finally, a further advantage of the frequency-domain approach is the gain in signal-to-noise ratio afforded by ATC over APCM of speech. We have shown previously [1] that taking the DCT of the residual appropriately rather than that of the speech signal achieves the same gain in signal-to-noise ratio.

3. TRANSFORM CODING

ATC of speech has been described in detail in the literature [3,4]. Here we review it briefly, mainly to point out the places where our implementation differs from that of others. An important step in ATC, bit allocation, requires a spectral model for the speech signal. The model we are currently using is given by

$$1/|H(w)| = 1/[|A(w)||P(w)|] \qquad (1)$$

in which $|A(w)|$ is the magnitude of the DFT of a 9th order LP inverse filter derived from the speech signal, and $|P(w)|$ is the magnitude of the DFT of a one-tap pitch inverse filter derived from the LP residual. Prior to bit allocation, the spectral model given in (1) is weighted to achieve a noise spectrum with a desired envelope given by $1/|A(w)|^{2\gamma}$, as is done in [4]. The bit allocation process is in fact quantization of the weighted spectral model of the speech signal on a logarithmic scale. It can be described by means of the following equations:

$$b_i = b_0 + [20 \log_{10}(A_i^{\gamma}/H_i)]/S , \quad 1 \le i \le N \qquad (2)$$

and

$$\hat{b}_i = \max\{0, \lfloor b_i + \beta \rceil\} \qquad (3)$$

such that

$$\sum_{i=1}^{N} \hat{b}_i = B \qquad (4)$$

where H_i represents the value of $|H(w)|$ at the discrete set of frequencies w_i, b_0 is the average number of bits per sample (B/N), N is the number of DCT points, B is the total number of available bits per frame, S is the quantization step size in decibels, b_i is the (fractional) allocated codelength in bits at frequency w_i, and $\hat{b}_i$ is the integer codelength corresponding to b_i. In (3), the symbol $\lfloor \cdot \rceil$ denotes taking the integer value nearest to the argument and the $\max\{\cdot\}$ function is used to prevent the allocation of negative codelengths. The adjustment constant β in (3) is varied iteratively until the condition in (4) is satisfied.

An important aspect of the quantization scheme given in (2) and (3) is the choice of step size S. Traditionally, the "6 dB per bit" rule has been used, i.e., S=6.02 dB. However, the increase in signal-to-quantization-noise ratio for the optimal $\hat{b}_i$-bit Max quantizers [5] used in quantizing the DCT coefficients ranges from 4.4 dB to 5.8 dB for bits in the range $0 \le \hat{b}_i \le 5$. Therefore, ideally, one should perform the bit allocation with unequal quantization steps. For simplicity, we have kept the uniform quantization in (2) with a compromise value of S=5 dB, and found this scheme to yield a higher signal-to-noise ratio than the case with S=6 dB.

In the system we are presenting here, we perform the bit allocation over the total signal bandwidth and transmit the DCT components of the fullband residual. The initial quantization accuracy is determined mainly by the total number of bits used at each frame, the step size S, and the value of γ for noise shaping. However, the accuracy and/or bandwidth of the received DCT components is further affected by the number of bits discarded by the channel. This last point is the subject of the next section.

4. EMBEDDED CODING

As mentioned in Section 2, the transmitter transmits at each frame a block of bits, which is divided into two major parts. The first part contains the bits representing the system parameters and the second part contains the bits representing the DCT components. It is assumed that the channel strips off bits starting at the end of a block, thereby discarding bits that represent DCT components. The codes representing the DCT components are arranged in a certain order prior to transmission. This ordering determines which bits get discarded. To study the tradeoff between the number of transmitted bits, the quantization accuracy, and the number of received frequency components, we investigated three ordering techniques. In all three techniques, to be described below, we assume that the receiver decodes the system parameters and performs the bit allocation as was done at the transmitter. This is

standard practice in ATC. In addition, we require that the receiver know how many bits are received each frame so that it knows where the next frame begins. This last piece of information is passed along by the channel itself.

The first bit-ordering technique we investigated is the simplest: the codes are arranged by order of increasing frequency. When the channel strips off bits from the end of each block, the high-frequency components are discarded first. The remaining codes represent a low frequency portion of the total bandwidth referred to as a baseband. As in a baseband coder, the receiver regenerates the missing high-frequency components. We use the method of high-frequency regeneration (HFR) by spectral duplication, which is explained in Section 5.

In the second ordering technique, the DCT codes are broken down into individual bits. The bits are then grouped together by order of significance, with the most significant bits in the first group and the least significant bits in the last group. To explain this grouping method, let us assume that the bit allocation produces codelengths between 1 and 5 bits. Therefore, the receiver expects to see 5 groups of bits. The first group contains the most significant bit of all 5-bit codes (in order of increasing frequency). The second group of bits contains the next most significant bit of all 5-bit codes and the most significant bit of all 4-bit codes. And so on, all the way to the last (5th) group of bits which contains the least significant bit of all the codes. When the channel strips off bits, the first bits to be discarded are the least significant bits of each transmitted code, resulting in decreased quantization accuracy. For example, a DCT component originally coded into 3 bits will be decoded by means of the 2-bit decoding table if its least significant bit has been dropped.

In the third ordering technique, the codes are grouped together according to their length. Thus, the receiver expects to see 5 groups of codes, with the first group containing all 5-bit codes, the second group all 4-bit codes, and so on, with the last group containing all 1-bit codes.

The second and third ordering techniques described above are more complex than the first, since they require arranging the data in the order of decreasing codelength. Informal listening tests have shown that the quality of the reconstructed speech when using the baseband-coder approach (the first ordering technique) is superior to that obtained by the second and third techniques. In general, our experience has been that the details of the low-frequency components of speech are perceptually more important than those at high frequencies. Thus, the task is to find a good compromise, at a given bit rate, between baseband width and quantization accuracy. At present, we feel that the first technique is giving the best overall speech quality for bit rates in the range 6.4 to 9.6 kb/s. We have not compared the three techniques at bit-rates above 9.6 kb/s. However, it is worthwhile pointing out here that we are seeking one single technique that will perform uniformly well over the whole range of data rates, because we are excluding the possibility of changing the coding algorithm while the system is in operation.

5. HIGH FREQUENCY REGENERATION

We have presented elsewhere [2] new HFR methods based on duplication of the baseband spectrum. In particular, we presented time-domain systems that perform spectral duplication by spectral folding and by spectral translation. We also presented a frequency-domain system [1] that performs HFR by spectral translation of the baseband. The principle of the method is based on the fact that, in the adaptive transform baseband coder, the baseband DCT components can be easily duplicated at higher frequencies to obtain the fullband excitation signal. Our present HFR method aims at duplicating, as closely as possible, the original fullband DCT of the residual, while accommodating the variable baseband width aspect of the present multirate system. The method is as follows.

The transmitter assumes a (fixed) nominal baseband width of 1000 Hz. Thus, the simplest spectral translation method would be to duplicate the region from 0 to 1000 Hz onto the regions from 1000 to 2000 Hz and from 2000 to 3000 Hz. In addition, to lock the high-frequency interval into place, by exploiting the quasi-harmonic structure of the speech spectrum, we shift the baseband around its nominal position and correlate it with the corresponding original DCT components that are in the region 1000 to 2000 Hz. The cross-correlation is done at the transmitter where the original DCT of the fullband residual is available. The same process is repeated for the next frequency band. Short lags from -3 to +4 spectral points are considered. (The total bandwidth is 128 points.) The optimal location is then chosen to be at the positive maximum value of the cross-correlation. Thus, we require an additional 3 bits of side information for each of the two high-frequency bands. The additional 6 bits are transmitted along with the system parameters.

At the receiver, the decoded baseband is translated up, starting at 1000 Hz (and 2000 Hz for the next band) and is moved further by a small amount as indicated by the 3-bit HFR code.

In practice, there are three deviations from the simple algorithm described above. The first is that the first few DCT components, starting at d.c. and up to half the pitch frequency, are not duplicated onto the high-frequency bands, nor are they considered in the correlation method described above. Second, we found that spectral flattening of the baseband at the receiver prior to HFR improves the speech quality. The third deviation from the algorithm is due to the fact that the received baseband is seldom equal to 1000 Hz. In fact, it varies from frame to frame. We have devised certain modifications to the method to deal with that problem appropriately.

6. RESULTS

We performed a large number of experiments to study the tradeoffs between all the interacting aspects of the system described in this paper. We summarize our results briefly below. The basic system is governed by an already existing ARPANET LPC vocoder operating with a frame size of 19.2 ms, i.e., 128 points, and transmitting 9 log-area-ratios each frame coded into 5,5,5,4,4,4,3,3, and 3 bits, respectively. Together with pitch and gain, the system parameters require a total of 48 bits. Thus, the basic LPC vocoder operates at a bit-rate of 2500 b/s. In addition, we require 4 bits for the pitch tap and 6 bits for the HFR codes, bringing the side information to 58 bits per frame. The remaining bits, which determine the total bit rate, are used to code the DCT coefficients. We investigated the following aspects of the system.

a. Initial Quantization Accuracy. To study the interaction between quantization accuracy and the number of received DCT components at a given data rate, we coded the fullband DCT at an average of 2, 2.25, 2.5, and 3 bits per sample. In our experiments, there was no maximum limit on the bit rate for the fullband case, although, in practice, the system will be operated at 9.6 kb/s or below.

b. Noise Shaping. We used various values of γ ranging between 0 and 1. Clearly, for γ closer to 1, the available bits are spread more evenly in the frequency range, resulting in a larger number of received DCT components at a given bit rate, at the expense of coarser quantization in the low-frequency region for voiced sounds.

c. Embedded Coding. We simulated the three ordering techniques described in Section 4 and evaluated informally the speech quality obtained with each. For each technique, we optimized the system in terms of the total fullband rate and the value of γ as in (a) and (b) above.

d. High-Frequency Regeneration. As described in Section 5, the transmitter assumes a nominal baseband width for which it computes the HFR codes. We investigated the use of 3-bit and 2-bit HFR codes, and the use of a nominal baseband width of 800, 900 and 1000 Hz.

For each choice of the above described parameter settings we tested the system at 9.6 kb/s, 7.2 kb/s and 6.4 kb/s, although, in principle, the data rate can be set by the channel to an arbitrary value. All tests were done with 5 male and 5 female sentences. Our present choice of a good compromise system operating uniformly well over the desired data rates is one where we code the fullband DCT of the residual at an average of 1.95 bits per sample, i.e., where the maximum bit rate is 16 kb/s. The value of γ is 0.9, and the embedded coding technique is the first (baseband coding), with a nominal baseband width of 1000 Hz and 3-bit HFR codes. The data rates of 9.6, 7.2 and 6.4 kb/s are achieved by keeping at each frame 124, 80, and 65 bits, respectively, out of the maximum total of 250 bits. The average width o the received baseband for the three cases is 140C 870, and 670 Hz, respectively.

At present, we feel that the above describe system is providing us with very good speec quality at 9.6 kb/s, good speech quality at 7. kb/s, and reasonable quality at 6.4 kb/s. Th problem at bit rates of 6.4 kb/s or below is tha the received baseband becomes too narrow, whic results in appreciable roughness in the code speech and some "thuds." Also noticeable at 6. kb/s, especially for female voices, is th reverberant quality of the reconstructed speech.

7. CONCLUSION

We presented in this paper a simple anc effective embedded-code multirate speech transfor coder. In going to a frequency-domain approach, we feel that we have accrued several advantages over time-domain coding schemes. Some of these advantages are: the ease with which spectral noise-shaping and quantization accuracy can be controlled, the ease with which the code can be embedded for multirate operation, and the ease with which high-frequency regeneration can be done for effective voice excitation. Our plans for the future include testing the use of a 3-tap pitch predictor, going to a 256-point block size, i.e., grouping two 19.2 ms frames together for transform coding purposes, and, in general, improving the speech quality at 6.4 kb/s. Finally, we also plan to implement the system on the FPS array processor AP-120B for real-time operation.

ACKNOWLEDGEMENT

This work was supported by the Advanced Research Projects Agency under contract No. F19628-78-C0136 and monitored by RADC/ETC.

REFERENCES

[1] M. Berouti and J. Makhoul, "An Adaptive-Transform Baseband Coder," Proceedings of the 97th Meeting of the Acoustical Society of America, paper MM10, pp. 377-380, June 1979.

[2] J. Makhoul and M. Berouti, "High-Frequency Regeneration in Speech Coding Systems," Int. Conf. Acoustics, Speech, and Signal Processing, Washington, D.C., pp. 428-431, April 1979.

[3] R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals," IEEE Trans. Acoustics, Speech, Signal Processing, Vol. ASSP-25, pp. 299-309, Aug. 1977.

[4] J.M. Tribolet and R.E. Crochiere, "Frequency Domain Coding of Speech," IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-25, pp. 512-530, October 1979.

[5] J. Max, "Quantizing for Minimum Distortion," IRE Trans. Inform. Theory, Vol. IT-6, pp. 7-12, March 1960.

APPENDIX F

A PRELIMINARY DESIGN OF A PHONETIC VOCODER BASED ON A DIPHONE MODEL

(IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, pp. 32-35, April 1980)

IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Denver, CO, April 1980

A PRELIMINARY DESIGN OF A PHONETIC VOCODER BASED ON A DIPHONE MODEL

Richard Schwartz, Jack Klovstad, John Makhoul, and John Sorensen

Bolt Beranek and Newman Inc.
Cambridge, MA

ABSTRACT

We report on the initial development of a phonetic vocoder operating at 100 b/s. With each phoneme, the vocoder transmits the duration and a single pitch value. The synthesizer uses a large inventory of diphone templates to synthesize a desired phoneme string. To determine a phoneme string from input speech, the analyzer takes into account the synthesis model by using the same inventory of diphone templates, augmented by additional diphone templates to account for alternate pronunciations. The phoneme string is chosen to minimize the difference between the diphone templates and the input speech according to a distance measure.

1. INTRODUCTION

There are many applications for digital speech transmission and speech playback that require very low data rates on the order of 100 b/s. In strategic communications, having a very-low-rate (VLR) capability would allow low power communications to avoid detection. Alternatively, it would allow sufficient power per bit to reliably "burn through" jamming networks. Under extreme atmospheric conditions or in highly shielded environments, only VLR transmission may be is possible. The same technology also supports speech storage for later playback using space comparable to that required for text. This would permit the storage of much larger amounts of speech in small devices than previously possible. A VLR vocoder would also allow the transmission of spoken messages using much the same mechanism now used for computer mail systems, without requiring high data rate connections to the computer and large amounts of file storage.

In this paper, we first describe a study performed to investigate the feasibility of a vocoder that operates at 75-100 b/s. After briefly discussing the results of this study, we report on the status of a current project involving the implementation of a VLR vocoder.

2. FEASIBILITY STUDY

In order to assess the feasibility of a phonetic vocoder, we undertook, in 1976, a short study in which we approximated the conditions of a 75-100 b/s phonetic vocoder [1].

The first question we asked ourselves was: "What should be the transmission units of a VLR vocoder?" We argued then that the transmission unit must be on the order of a phoneme. At an average speaking rate of about 12 phonemes per second, simply transmitting phonemes requires about 60-75 bits. Adding just the barest amount of intonation information (3 bits total for each phoneme duration and a single pitch value) brings the rate up to 100 b/s. Even if the transmission units used by such a vocoder were not phonemes per se, the spectral difference between any two units would necessarily be on the order of the difference between phonemes. Hence, errors in such a vocoder would directly cause a loss of intelligibility, since changing a phoneme could change the meaning.

The basic phonetic vocoder that we

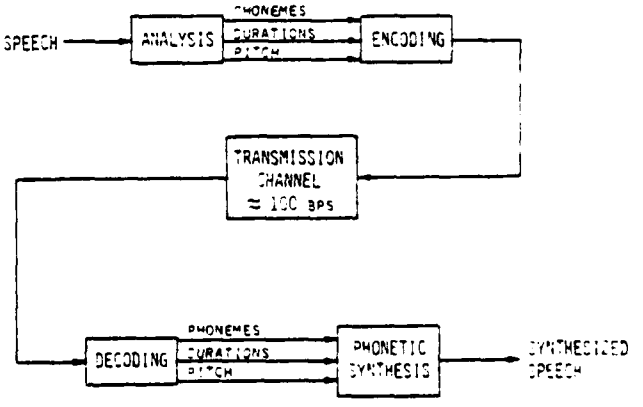


Fig. 1 Very low rate (VLR) Phonetic Vocoder.

implemented is shown in Fig. 1. Using the Acoustic-Phonetic Recognizer (APR) component of the HWIM Speech Understanding System [2] as the phoneme analyzer, we produced a set of phoneme strings with associated durations and pitch values to be transmitted to a phonetic synthesizer. The transmitted pitch values specify a piecewise linear function (linear during each phoneme). The function is determined by minimizing the weighted squared error between the linear function and the original pitch track. The synthesizer used was Dennis Klatt's synthesis-by-rule program [3]. The test was not performed under optimal conditions; neither the analyzer nor the synthesizer were designed for phonetic vocoding, nor were they completely compatible with each other. However, we felt that the results derived would still be useful.

To start, we chose a subset of sentences on which the APR performed better than average, with phoneme recognition rates ranging from 65%-95%. A two-way conversation, in which one side was vocoded by this simulation system and the other was a natural voice, was played to a panel of listeners. The panel was asked to transcribe the vocoded side of the conversation. The primary result was that sentences in which 80% or more of the phonemes were correctly recognized by the APR were transcribed with little difficulty by the panel. Those sentences that contained more errors were largely unintelligible. A second important result was that a very rough phoneme duration, and a heavily quantized single pitch value for each phoneme, were sufficient to preserve the original intonation.

We concluded, therefore, that for a phonetic vocoder operating at 100 b/s to be useful, it must have a phonetic recognition rate of at least 80% and natural phonetic synthesis. Below, we describe our first effort at designing such a system.

3. DIPHONE VOCODER

We have recently begun working on a phonetic vocoder that is designed to transmit intelligible speech at an average rate of about 100 b/s. The block diagram for this vocoder is the same as for the experimental system shown in Fig. 1. The vocoder extracts and transmits a sequence of phonemes. It also transmits the phoneme durations and a single pitch value for each voiced phoneme in order to preserve the intonation in the input speech. However, the phonetic analysis and synthesis components of this vocoder have been changed from that of the experimental system. The basic unit that we have chosen for use in both the analyzer and synthesizer is the diphone. A diphone is defined as the region from the middle of one phoneme to the middle of the next phoneme. The diphone is a natural unit for synthesis because the coarticulatory influence of one phoneme does not usually extend much further than half way into the next phoneme. Since diphone junctures are often at articulatory steady states, minimal smoothing is required between adjacent diphones. Also, the difficult task of representing phoneme transitions by acoustic-phonetic rules is avoided.

Below, we describe the diphone synthesis and diphone analysis components of the phonetic vocoder. At the present time, both programs are being designed for a single speaker.

3.1 Diphone Synthesis

Since the design of the analyzer depends critically on the synthesizer model, we shall discuss the synthesizer first. Fig. 2 shows a block diagram of

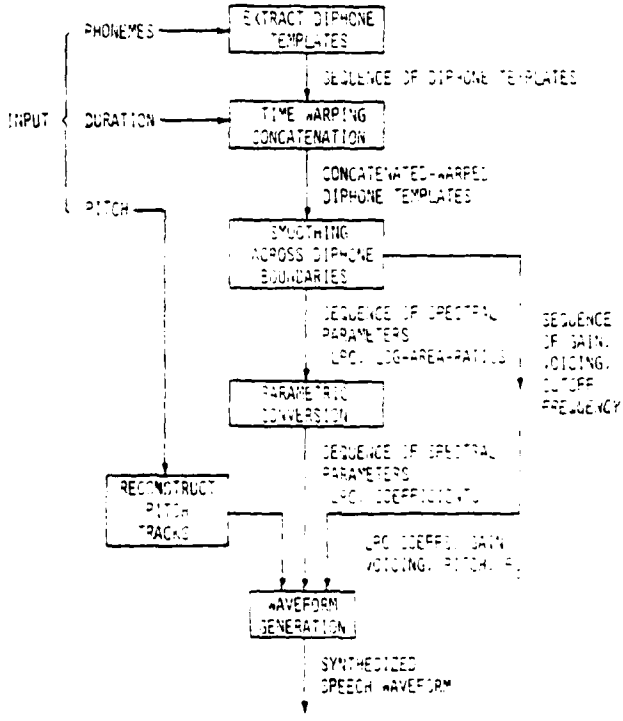


Fig. 2 Diphone Synthesis method.

the synthesis program. The phonetic synthesizer [4] uses the transmitted phonemes, durations and pitch values, to produce a sequence of control parameters (LPC parameters, voicing, pitch, gain, cutoff frequency [5,6]) for an LPC synthesizer. These parameters are stored

in a large inventory of templates - one for each diphone. A template contains, for each 10 ms frame, the LPC parameters and a value of gain. The LPC parameters are stored as Log Area Ratio (LAR) parameters, since these are used directly by the diphone synthesis program. The diphone inventory has been selected to account for many of the stronger coarticulation effects in English. In particular, the diphone template inventory is chosen to differentiate between prevocalic and postvocalic allophones of sonorants, to account for changes in vowel color conditioned by postvocalic liquids, to allow exact specification of voice onset time, and to permit the synthesis of glottal stops, alveolar flaps, and syllabic consonants. The diphone templates are extracted from a carefully constructed set of short utterances. The current total diphone synthesis inventory is approximately 2650 diphones.

The first step in the synthesizer is to translate the input phoneme sequence into a diphone sequence. In some cases, there are multiple diphone templates for a single phoneme pair. The template used is determined by the surrounding phonetic context. Each of the diphone templates is warped in time so that the input phoneme duration requirements are satisfied. The time-warping takes into account the relative inelasticity of the phoneme transition region. Next, the program smooths between consecutive diphone templates to minimize gain and spectral discontinuities. The smoothing algorithm is designed to preserve the original parameter tracks intact where possible. Continuous pitch tracks are reconstructed by linear interpolation of the sequence of single pitch values, one for each phoneme. The cutoff frequency and voicing flag for each frame are determined by rule from the phonemes being synthesized. The resulting continuous parameter tracks (specified every 10 ms) are used to control an LPC speech synthesizer.

3.2 Diphone Analysis

The analyzer takes into account the synthesis model by using a network of diphone templates to recognize the sequence of phonemes in the input speech. The diphone network consists of nodes and directed arcs. An example of a simple network is shown in Fig. 3. There are two types of nodes: phoneme nodes and spectrum nodes. The phoneme nodes (shown as labelled circles) correspond to the midpoints of the phonemes; there is one such node for each phoneme. These phoneme nodes are connected by diphone templates. Each diphone template is represented in the network as a sequence of spectrum nodes (shown as dots). When two or more

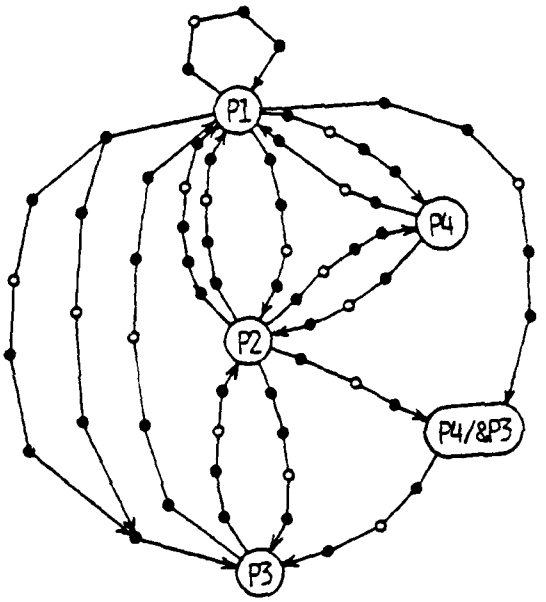


Fig. 3 Example of Diphone Template Network for Four Phonemes.

consecutive spectra in the original diphone template are very similar, they are represented by a single spectrum node in the network. The open dots indicate the first spectrum node in the original diphone template that is at or past the labelled phoneme boundary. Note that, in Fig. 3, the diphone template P1-P2 is distinct from the template P2-P1. Also note the possibility of diphones of the type P1-P1. The network allows for two or more templates going from one phoneme to another (e.g., P2-P1). Branching and merging of paths within a template is also allowed (e.g., P1-P3). Finally, the network allows the specification of diphones in context. The node P4/&P3 represents the phoneme P4 followed only by P3. Thus the template P2-P4/&P3 can be different from the unconditioned template P2-P4. The generation and training of the network is discussed further in Section 3.3. The analyzer chooses the sequence of templates that best matches the input speech according to a distance measure. Since the received speech is spectrally close to the original it is hoped that this procedure will suffer minimally from phoneme recognition errors.

The network matcher uses a dynamic programming algorithm which attempts to find the sequence of templates in the network that best matches the input. The basic operation of the program begins by updating each "theory" by the addition of the newest input frame. A theory consists of a detailed account of how a sequence of input frames is aligned with the network,

along with a total score for that correspondence. Since the program allows more than one input frame to be aligned with a single spectrum node, each theory is replaced by at least two new theories: one that corresponds to the new input frame being aligned with the same node as the previous frame, and one for each of the nodes immediately following that most recent node. The program then discards all but the best n theories, where n is fixed.

To decide on the phonemes to transmit, the program examines all the remaining theories after updating each theory with the newest input frame. If all the theories have a common beginning, the phonemes corresponding to this common beginning are transmitted. At this early stage in the research, we find that preserving a few hundred theories for each frame essentially guarantees that the program will find the globally optimum path through the network. As the template statistics are updated through training, we hope that the number of theories required for this "bounded breadth search" will decrease somewhat. Thus, the analyzer requires a variable lag between the input and the transmitted output. The program can also be given a maximum delay parameter which forces a decision if the transmission delay exceeds a threshold. These two features allow the analyzer to operate in a continuous mode, rather than on one sentence at a time.

The spectral distance measure currently used by the analyzer in this vocoder is a simple weighted Euclidean distance between the Log-Area-Ratio (LAR) vectors of the template and the input. Other distance measures will be used as needed.

3.3 Network Generation and Training

Initially, the network is generated by duplicating all the templates used by the diphone synthesizer. Then, the network is augmented by the addition of new paths as a result of extensive training. This allows the network to represent a variety of pronunciations for each diphone.

The analyzer allows the researcher to train the network on new speech interactively. The researcher guides the matcher through the correct phoneme sequence. Once the program has aligned the input speech with the network, the researcher then instructs the program to use the aligned speech to update the templates involved. If the input is substantially different from a region of the network, the program can add new paths corresponding to partial or complete templates. In this way, the network will eventually contain a variety of different acoustic realizations for each diphone.

4. SUMMARY

We have described the initial experiments and current work leading to the development of a very-low-rate phonetic vocoder that operates at around 100 b/s. The diphone synthesizer for a single speaker is complete and, given the correct phoneme sequence, produces highly intelligible speech. An initial version of the diphone analyzer has been designed and implemented, but will need extensive training before conclusions can be drawn.

Finally, once the analysis component of the VLR vocoder is operating at a level of performance sufficient for intelligible communications, it would also likely lead to the design of more advanced automatic speech recognition systems.

ACKNOWLEDGEMENT

This work was supported by the Advanced Research Projects Agency under contract No. F19628-78-C-0136 and monitored by RADC/ETC.

REFERENCES

[1] J. Makhoul, C. Cook, R. Schwartz, and D. Klatt, "A Feasibility Study of Very Low Rate Speech Compression System," Report No. 3508, Bolt Beranek and Newman Inc., Cambridge, MA, Feb. 1977.

[2] W.A. Woods et. al. (1976) "Speech Understanding Systems, Final Report: Vol. II," BBN Report No. 3438.

[3] D.H. Klatt, "Structure of the Phonological Component for a Synthesis-by-Rule Program," IEEE Trans. ASSP-24, pp. 381-398, 1976.

[4] R. Schwartz, J Klovstad, J. Makhoul, D. Klatt, and V. Zue, (1979) "Diphone Synthesis for Phonetic Vocoding," Proc. IEEE-ICASSP, April 1979, pp. 891-894

[5] J. Makhoul, R. Viswanathan, R. Schwartz, and A.W.F. Huggins, "A Mixed-Source Model for Speech Compression and Synthesis," 1978 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Tulsa, OK, April 10-12, 1978, pp. 163-166.

[6] A.W.F. Huggins, R. Schwartz, R. Viswanathan and J. Makhoul, "Subjective Quality Testing of a New Source Model of LPC Vocoders," presented at the 96th meeting of the Acoustical Society of America, Nov. 1978, paper GGG12.